OULUN AMMATTIKORKEAKOULU

Katja Saapunki

ROOT CAUSE ANALYSIS AND ESCAPE DEFECT ANALYSIS IMPROVEMENT AT CONTINUOUS DELIVERY BY DATA-DRIVEN DECISION-MAKING

ROOT CAUSE ANALYSIS AND ESCAPE DEFECT ANALYSIS IMPROVEMENT AT CONTINUOUS DELIVERY BY DATA-DRIVEN DECISION-MAKING

Katja Saapunki
Master Thesis
Spring 2023
The Degree Programme in Data Analytics and Project Management
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Data Analytics and Project Management

Author(s): Katja Saapunki
Title of the thesis: Root cause analysis and escape defect analysis improvement at continuous delivery by data-driven decision-making
Thesis examiner(s): Ilpo Virtanen
Spring 2023 Pages: 65 + 3 appendices

This thesis was done to software development organization of global telecommunication vendor. The main driver for this thesis was the ongoing transition of continuous integration (CI) and continuous delivery (CD) practices at the telecommunication sector. It challenges current software development methods by fast delivery cycles and high software quality requirements. One of the impacted areas in that evolution is root cause analysis (RCA) and escape defect analysis (EDA).

The main targets of this study were to identify pain points in company's current RCA/EDA practices and to define ways how RCA/EDAs could be left-shifted in software development life cycle and that way better support CI/CD to fulfill customer expectations.

The research was a mixed method study with embedded design. Quantitative data was collected from the company's internal tools and databases. Qualitative data collection was done by interviewing company workers who are involved in RCA/EDA work and by taking part in company's internal workshops related to RCA/EDA. Data analysis and visualization was done in Power BI and results combined both quantitative and qualitative data, nevertheless the bulk was quantitative research.

According to analysis main pain points of current RCA/EDA practice were related to the long lead time of RCA/EDA, high ratio of completed analysis without any preventive actions named or implemented and imprecise RCA/EDA targeting, which caused software faults leakage to customers.

To support CI/CD better, improve customer experience and manage effort and costs more efficiently, three main enhancements were proposed:
1. RCA/EDA decisions for internal defects to be managed via automated data-driven decision making with customer impact included.
2. Metrics to support priorization and offer visibility to the whole pipeline starting from defect reporting and ending to implemented preventive actions.
3. Use RCA as a tool for finding root cause of topics which went well to enable learnings from successful practices and processes at organization level.

Machine learning opportunities in software fault handling and RCA/EDA work was not considered in this study, but it was seen as an opportunity for further research.

Keywords: data-driven decision-making, root cause analysis, escape defect analysis, RCA/EDA, continuous integration, continuous delivery, CI/CD, fault handling, fault management, software development, agile

CONTENTS

ABBREVIATIONS

AI	Artificial intelligence
CD	Continuous development / delivery / deployment
CI	Continuous integration
CI/CD	Continuous integration and continuous delivery
DDDM	Data driven decision making
EDA	Escape defect analysis
FR	Fault report (defect)
I&V	Integration and verification
JSON	JavaScript Object Notation
RCA	Root cause analysis
REST API	Representational state transfer application programming interface
R&D	Research and development
SCM	Software configuration management
SDLC	Software development life cycle
SW	Software
TL9000	Quality management system standard for international telecommunications industry
5Why	Iterative technique to explore the cause-and-effect relationships underlying a particular problem

1 INTRODUCTION

Telecommunication industry is moving towards continuous integration (CI), continuous delivery (CD) and continuous deployment (CD) practices in software business area. That transition challenge telecommunication equipment and software manufacturers as customers expect shortened delivery cycles and improved software's quality (Silverthorne 2022).

One practice used by software development organizations to improve a product's quality, is the root cause analysis (RCA) and escape defect analysis (EDA). Process is called RCA/EDA. It holds set of tools and techniques used to analyze the root cause of the identified problem, find out why the defect slipped from organization's internal quality assurance to customer and define actions how to avoid the same problem happening again in the future. (Tableau Software LLC.)

The purpose of this thesis is to find out key ideas how RCA/EDA work can be embedded as early as possible as a part of software development pipeline to support continuous delivery and deployment practices in the best feasible way. The target is that software product's problematic areas, which need more attention, will be found automatically already during the product's development phase. So that software faults are corrected, needed improvement actions identified and implemented already before product is delivered to customers.

The traditional way to execute RCA/EDA is to wait for software fault report or escalation from the customer, correct the defect and after that trigger RCA/EDA actions based on some pre-defined, usually fixed rule. This means RCAs are typically handled reactively, not proactively (DiFrancesco 2021). To reach requirements of continuous delivery and meet customer expectations, left shifting of RCA/EDAs becomes essential. The target is to get RCA/EDAs left-shifted by the support of more automated and data-oriented decision-making practice compared to current reactive model. This will also bring cost savings, as the earlier fault is detected, the more cost-efficient the problem-solving is (Churchwell 2018).

1.1 Case Company

Case company in this research is one business group from a large network equipment manufacturer. Main products of the manufacturer are related to telecommunication infrastructure, both software and hardware. This study concentrates on one of the biggest business groups in the company. Business group is a provider of fixed and wireless network equipment and services. Customers of that business group are global network operators and service providers. Software development process and practices are in the content of this study, hardware is excluded.

To fulfil expectation of telecommunication industry customers, the company is on the path to fully adopted CI/CD practices. There are agile software development methods and lot of automation in place already in company's software development and delivery area. Continuous integration, testing and delivery model is under deployment with selected customers. At the same time the shorter delivery cycle challenge company's existing software development processes, like RCA/EDA.

Expected outcome by the case company is, that this thesis identifies pain points of company's current RCA/EDA process, defines list of most urgent improvement actions needed to get RCA/EDAs to better support continuous deliveries and specifies a dashboard which automatically identifies and visualizes areas need RCA/EDAs as early as possible during software product's development lifecycle to meet customer expectation towards software quality and continuous deliveries.

CI/CD processes and practices as such are not expected to be improved by this thesis but left-shifting RCA/EDA as a part of the company's software development lifecycle as early as possible requires also considering and understanding CI/CD way of work.

1.2 RCA/EDA in case company

RCA/EDA process' input is a software fault correction. After software defect's root cause is found and fix done, RCA/EDA process starts. By default, all corrected software defects will get a light RCA. It is a top-level analysis with two outputs: identified root cause and in which software development phase the fault should have been originally detected. Light RCA is done immediately after

the root cause of the fault has been found, software fix done and successfully verified by the defect's reporter.

On top of the light RCA, the complete 5Why-based RCA/EDA is done for selected defects reported either by customer or by the company's internal testing teams. Selection criteria is defined in company's RCA/EDA process. Criteria is based on defect's TL9000 problem severity classification (Appendix 1) and other parameters, like fault's impact and criticality towards product, customers and company. Evaluation of RCA/EDA selection criteria is done by process owner in co-operation with experts. After fixed rule for the product is defined and agreed, triggering RCA/EDA tasks starts for all new fault reports meeting that criterion. An analysis can be also requested for any other defect even not meeting the actual defined criteria if there is for example a business reason to request detailed analysis and preventive actions. The actual task creation to the RCA/EDA database is done by in-house tool either automatically or via manual triggering. For each main RCA/EDA task in the database, several subtasks can be created. The detailed structure of RCA/EDA backlog and its relations is defined in Appendix 2.

By default, light RCA and 5Why RCA/EDA both concentrate on individual defect, its root cause, why the defect was not detected in earlier software development phases and how to prevent that defect to re-occur in the future. In company's RCA/EDA process there is also possibility to execute a bulk RCA/EDA. It is an analysis, which gathers bigger set of similar defects together and concentrates in creating one common RCA/EDA with a wider scope. Data for bulk analysis is received from light RCAs and 5Why RCA/EDAs.

Customer reported defect will always generate a request for a special customer fault analysis report. It is delivered to customer based on 5Why RCA/EDA results. There is an inspection executed and quality grading given for each RCA/EDA. Customer can take part to customer fault analysis report review.

All improvement items defined by analysis are collected into improvement backlogs of R&D teams. One RCA/EDA can produce improvement actions towards several separate R&D teams. How and when improvement actions are really implemented and deployed, depends on each team. They have freedom to target all improvement actions according to their own priorities and available resources.

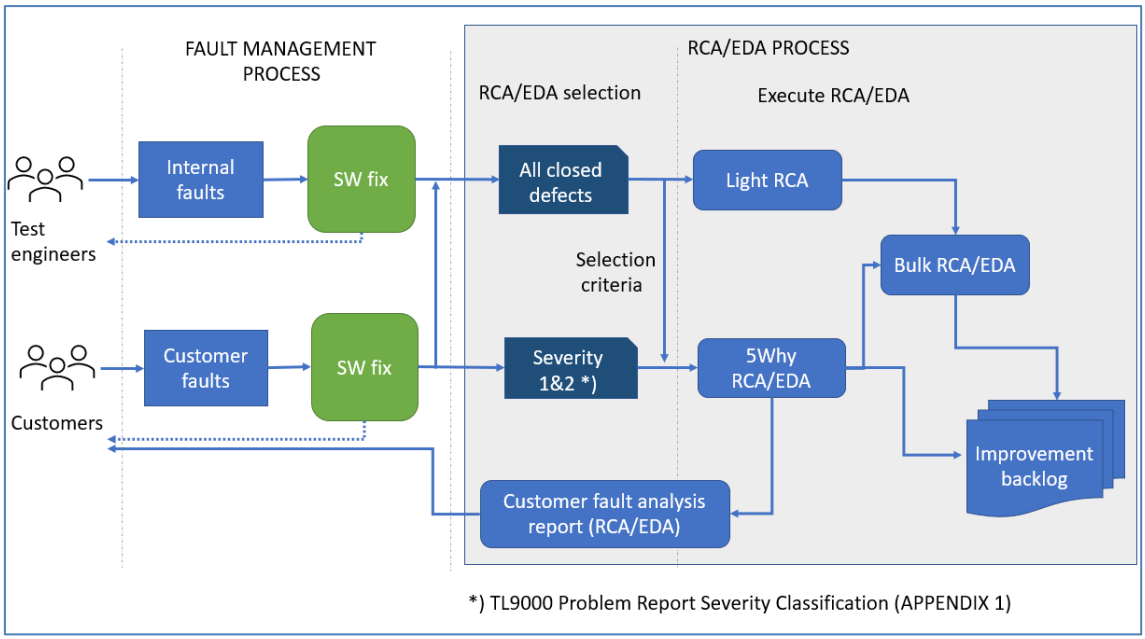


Figure 1. RCA/EDA process in case company

RCA/EDA is not part of the company's fault management process, but it is completely independent process with own defined roles and responsibilities. Actions are covered by dedicated, nominated RCA/EDA responsible persons. Analysis is triggered after the defect's root cause is identified and fix to software done. Both analysis and improvement actions are recorded in one common backlog. There is no visibility of RCA/EDA requests or actions in company's fault handling system and tool, all tasks related to RCA/EDAs are handled with an own tool.

There are several measurements in use to follow RCA/EDA process and its success. Basic statistic "5Why RCA/EDA cycle time" measures how long it takes to complete analysis for customer originated cases, as well RCA/EDA coverage among all fault reports. Improvement actions defined by analysis are also followed by own execution metric. Root causes and escape causes are collected and categorized to be used as a base for future improvements. All measurements with source data are available for all company workers.

1.3 Research questions and method

Objective of this thesis were defined with following research questions:

1. What are main pain points in case company's current RCA/EDA process, practices, and tools?

2. What improvements needed to RCA/EDA process, practices, and tools to support continuous deliveries?
3. How to left-shift RCA/EDA work to avoid customer escalations?

The research strategy was mixed method with embedded design. In this research both quality and quantitative research methods were used. Quantitative data was collected from company's existing, internal database sources like RCA/EDA, feature, integration and verification (I&V) and fault management tools. Data visualization was done in PowerBI. Data was imported to PowerBI via "rest_api" in JSON format. All databases used in this thesis supported data export.

Simultaneously, to reach more insights on current practices, pain points and views for future, qualitative data was also collected. It was done via structured interviews. Interviewees were selected based on their job role: software developers, testing engineers, RCA/EDA practicians and different level managers. On top of interviews, to wider understand the problem in hands, details were also gathered by taking part in company's internal workshops related to RCA/EDA improvements.

Research results are a combination from quantitative data analysis and qualitative analysis done based on interviews and workshops. As quantitative data had larger design in this research, qualitative data is considered as secondary.

1.4 Structure of the thesis

This thesis report is divided into six chapters. After Introduction, the next chapter presents background studies done via reviewing literature connected to thesis topics. The chapter presents the current state of data-driven decision-making (DDDM) and RCA/EDA practices. On top of current state, it also covers trends, views, and predictions about the future of RCA/EDAs when continuous integration (CI) and continuous delivery (CD) methods are fully deployed and in everyday use in an organization.

Chapter three concentrates in the data collection methods used in this thesis. The chapter contains a detailed description of used datasources and tools. On top of that, also data formatting, validating, modelling and visualization principles are described.
The actual results of the analysis are presented in the chapter four. Graphs, charts and tables are used to visualize analysis results. Within analysis part, results combined from quantitative and qualitative research are presented.

In chapter five findings from data-analysis are discussed. Results are handled through each research question. Background studies and theory behind are combined with analysis results in discussion. At the end of the chapter, research process, known limitations and future research suggestions are described.

The last chapter is a conclusion part, which summaries key findings and results of this research.

2 BACKGROUND STUDIES

Data plays critical role in today's rapidly changing environment. It helps organization to streamline operations and finetune effective usage of resources. Analysed data offers leaders a view for areas needs optimization, like processes to be improved, waste removal and customer satisfaction improvement. (Ismail 2017.) The main target of background studies chapter is to define the theoretical framework by gathering knowledge from existing literature. The chapter explains how data can be utilized and used to support organization to find out problematic areas on software development as early as possible and to make corrective actions before software is delivered to customers.

Three main subjects were selected as a content of the background study since these areas were perceived to have the most significant impact in improving software development practices and in answering to research questions set in this thesis. Selected areas for literature review are data-driven decision making (DDDM), RCA/EDA practices and CI/CD methods.

As the target of this thesis is to find out ways how current RCA/EDA can be left-shifted in software development's lifecycle by support of DDDM, the existing literature of DDDM is reviewed in the first section of the background study. RCA/EDA current practices are reviewed and presented as a second topic. On top of today's status, literature review also contains trends and views about the future of DDDM and RCA/EDA. Review will introduce predictions how shorter delivery cycles and continuous software development methods challenge existing way of work. It will also offer a view how complexity of systems and organization effects on the topic. Third part in the background studies covers CI/CD methods. Despite improving CI/CD practices as such is not in the context of this thesis, it plays a vital role as a key driver why improvements to current practices are needed. CI/CD basic principles and terminology relevant to the topic are covered as a part of theoretical framework.

2.1 Data-driven decision making

In today's modern world data is everywhere. According to estimation by Statista, 79 zettabytes data was generated in 2021 worldwide. The forecast for 2025 is that global data production will be duplicated to 180 zettabytes. (Taylor 2022 a.) Nowadays data is mostly reachable on-line, and people are used to find answer to their questions in seconds from global data storages. This challenges companies as the same is expected also in work related questions. Typically, data contains the history and has been used to understand what happened in the past and why. But this level of analysis in no more enough to fulfil requirements of today's fast evolving business needs. Instant data shall be available and used as a base of decision making. (Anderson 2015, chapter Preface; Tunguz & Bien 2016, chapter 1.)

Effective data-driven decision making requires online data availability for decision makers. Pykes (2022) summarizes in his blog the importance of data handling and analysis for companies. The real value and benefit of data is gained, when data is gathered rapidly, completely, accurately and is connected to other relevant data. He states there are five key areas which cannot be any more performed in modern companies without instant, accurate data. Those areas are decision making, problem solving, understanding performance, improving processes, and understanding customers. (Pykes 2022.)

Term data-driven decision-making is explained by Miller (2019) as a decision-making process, where patterns and facts are extracted from the collected data. Those facts are utilized to influence decision-making. When organization has data-driven decision-making practice in use, it means decisions are done based on actual data and collected facts instead of pure intuition or observations. (Miller 2019.)

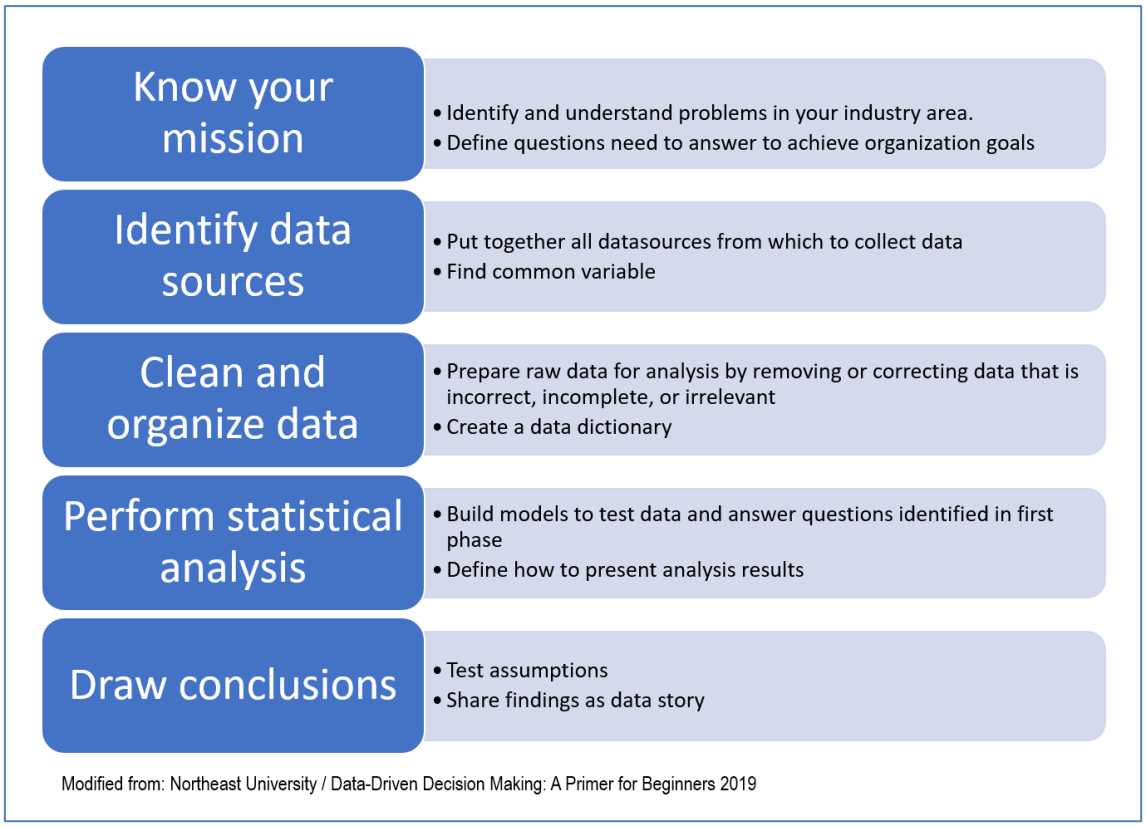


Figure 2. Data-driven decision-making phases (Miller 2019)

Anderson (2015) goes deeper with his definition of data-driven decision-making. He confirms that the decision making based on data requires analysis, shared reports and dashboards. But just generating those and having them available, is not enough to call the organization as data driven. Reports and dashboards often tell what happened in the past, but they do not contain the explanation why something happened. Predictions and recommendations shall be in place at the data story and the most importantly, organization's culture must contain in-built attitude to really react based on facts shared in reports. (Anderson 2015, chapter Preface, chapter 1.)

Companies which have fully adopted DDDM, have 5 - 6% higher output and productivity compared to organizations which do not follow DDDM practices. The equal difference is seen also when taking into consideration other investments and level of information technology usage in companies. (Brynjolfsson & Hitt & Kim 2011, chapter Abstract, chapter Results and Discussion.) Despite clear increase in productivity seen, especially organizations where agile software development methods with short delivery cycles are in use, have still unfulfilled potential of data-driven decision-making like explained in the study by Berntsson Svensson & Feldt & Torkar (2019, 69, 71). Learning from customers, short feedback cycles and customer view are crucial factors in agile software development where CI/CD with short delivery cycles also takes place. It is important to have a fast data

based decision-making process in use to ensure customer expectations for CI/CD are met. (Berntsson Svensson & Feldt & Torkar 2019, 69; Santos Jr et al. 2021, chapter Introduction.)

One key item preventing agile software development organizations to use full potential of DDDM, can be separate applications companies use to support agile software processes. Applications store useful data for decision making, but data is divided into heterogenous applications with different behavioral models and focusing only to specific software process areas. This can lead to conflicts and misunderstandings. Another issue which blocks DDDM at agile organizations is that deployment of DDDM is never allowed to create any bottleneck to the process natural agility. (Santos Jr et al. 2021, chapter Introduction.) Agile software development's context does not allow any waste, like additional waiting times, multitasking on different uncompleted tasks or any actions without value added (Kim et al. 2016, part 1, chapter 2; Bernstein).

In telecommunication sector, every second company used DDDM practices in 2020 according to survey made by Statista (Taylor 2022 b). In TechnoVision report (Capgemini 2023), the future of DDDM trends is considered to continue as predictive. According to report the next step in DDDM is to further improve efficiency and productivity. It will come by automating the decision making and producing real-time available predictive reports with help of artificial intelligence (AI). (Capgemini 2023; Sevilla 2023.) DDDM's unfulfilled potential especially in agile software development could be improved by adding more intelligence and automation to data collection, analysis, visualisation and connecting relevant data to current agile decision-making processes (Berntsson Svensson & Feldt & Torkar 2019, 83). The next level's innovative technology solutions can be delivered more efficient way with smaller resources when investigating and adopting latest trends and possibilities of AI, intelligent automation and self-optimizing technologies. In practice the more intelligence to be added to DDDM, the lower amount of highly specialized people is needed in product development. The expectation is smaller environmental impact and costs. (Capgemini 2023.)

2.2 Root cause analysis (RCA) and escape defect analysis (EDA)

Root cause analysis (RCA) process including escape defect analysis (EDA) is considered as a main problem-solving method concentrating to discover primary and secondary root causes of incident, problem, or event. There are three goals defined for RCA. The primary goal is to identify the root cause. The second goal is to gather wider understanding about the issue i.e., how to fix

the problem, either with a solution or a workaround, and to identify other possible underlying issues related to the root cause. The third goal is to collect learnings from the analysis and define actions to prevent the same issue to happen again in the future. (Etti-Williams 2017; Tableau Software 2022; Hammad 2020 a.)

The key driver in companies to execute RCA is the thought, that it is more effective to handle problems with systematic, planned preventive actions and solving hidden issues compared to just cover ad hoc symptoms and extinguishing fire (Tableau Software 2022). Effective RCA is a valuable tool to any organization, but especially large companies benefit for executing RCAs. In big organizations there are many issues always arising and without a solid plan how to deal with them, organization can become totally reactive. (Latino & Latino & Latino 2011, 1.)

In software organizations systematic RCA is seen as a method to improve software quality and processes. Completed RCAs will eventually lead to organization level productivity and efficiency enhancements and better customer experience by reducing number of software defects in the product. (Hammad 2020 a; SoftwareTestingHelp 2023.)

2.2.1 RCA/EDA process and execution

The usage of root cause analysis began more than 60 years ago, when engineers noted the need to find out more systematic ways and methods to improve the troubleshooting of defective products. During decades RCA has been adopted in almost all industry and scientific areas. And no matter what the area is, the analysis target is always to search for the underlying cause of a quality problem. Related to the problem currently in hands and its area, RCAs can be divided in to four high-level categories: production-based, safety-based, process-based, and systems-based. As there are so many diverse types of RCAs existing, there are also several methods and tools which can be used to execute RCA (Table1). The used RCA technique shall be connected to the problem under investigation, but ideally, it is always a combination of scientific methods and the necessary tools for collecting and analysing data as well generating and evaluating hypothesis. (Barsalou 2014, 51; Edel 2016, chapter 3.)

Table 1. Most common root cause analysis techniques and strategies (Tableau Software 2022; Hammad 2020 b).

Technique	Description	Output
5 Why approach	Consecutive question "Why?". Approximately after 5th "Why?" concise reason found.	Last "Why?" leads to failed process, which can be fixed.
Fishbone/Ishikawa diagram	Brainstorm causes and effects, visualized format as spine of the fish.	Categorised source(s) of the issue, most likely root causes identified. Unrelated categories eliminated.
Change analysis	Determine what, where, how, and extent of the problem. Compare with situation where defect is not present.	Identified reason(s) lead to defect.
Event analysis	Analysis of events which lead to incident.	Identified causal and contributing factors possible for each event.
Failure Mode and Effects Analysis (FMEA)	Analysis of system processes / products.	Prioritized list of identified failures in process of the system or design of the product.

Preferred way to perform RCA/EDA work is to use a focused, cross functional team approach. In case the issue is complex, a larger and more formal team shall be established to cover RCA/EDA actions. Analysis task always need commitment, focus, effort and usage of scientific methods and quality tools. (Hammad 2020 c; Barsalou 2014, 57-58; Dunn.) That is why RCA/EDA team setup in organization using agile software development shall be planned well, as non-planned and uncontrolled interruptions are violating agile development method rules. One reason for interruption can be a request to analyse identified defect in earlier, even long time ago implemented software, so

planning needed to mitigate and minimize RCA/EDA work's impact to new feature development. (Berteig 2020.)

According to Weeks (2018) and Hammad (2022 c) agile development practices combined with large software development organization can cause also other disadvantages in execution of RCA/EDA process. One problem is that RCA/EDA does not necessary provide enough complexity for large software systems as it is focusing by default only to one root cause at the time when in complex system there can be interconnections and patterns which are unknown by the RCA/EDA team. Another issue is the blaming nature of RCA/EDA, which can undermine a positive agile culture. Analysis often concentrates only on negative things happening within the organization. (Weeks 2018; Hammad 2022 c.) It is important for RCA/EDA success, that all actions include blameless objectivity. Output of RCA/EDA process shall be always formatted as actionable steps without direct pointing to an individual or a team. It will increase the commitment of the organization and decrease the resistance towards analysis tasks. Teams are more likely to implement proposed actions when blaming is removed from reports and recommendations of RCA/EDA. (Bigelow 2021 a.)

Barsalou (2014, 45) reminds in his book that after each RCA/EDA, also lessons learnt shall be considered. It includes for example analysis of possible other improvement opportunities into product or process. Lessons learnt shall be recorded and shared within the organization. In small companies this is rather easy, in large multinational company it is more complicated. It is most useful to collect lessons learnt when target organization is a large company that develops the same software components in several locations and with same processes. Crucial part is to ensure the entire organization becomes aware of learnings and improvements. (Barsalou 2014, 45.)

2.2.2 Proactive and predictive RCA

There is underestimated potential in RCA process usage in organizations. For example, it is not always needed an existing defect to trigger an analysis. RCA/EDA can be also considered as an effective tool to find out root cause of a success. When team notice good things in organization happening, like overachievement or reaching early deadline, it is not a bad idea to find out the root cause of why things went well. Analysis why some processes are performing well, can help to prioritize and protect key factors. Identifying well performing processes and reason for that, can be

used as a source of lessons learnt and continuous improvement all over the organization. (Tableau 2018; Hammad 2020 c.)

The latest literature notifies also predictive and proactive usage of RCA/EDA methods. Etti-Williams (2017) uses the term Potential Problem Solving (PPS) in his definition of predictive RCA/EDA. Even it is originally considered only a defect's investigation tool, it is also highly solution and problem prevention-oriented method. The main question in predictive analysis is: "What could go wrong?". Pro-active RCA can be considered as an analysis, which is executed before any failure or defect yet slipped from the product's development team to further phases. Analysis is used to define preventive actions, implement those, and avoid defect from occurring in the future. (Etti-Williams 2017; Hammad 2020 d.) RCA is an interactive process which has been nowadays used by organizations also as a tool for continuous process improvements. Companies have adopted RCA methods for forecasting occurrence ratio of defects which have not yet even appeared. (Hammad 2020 d.)

2.2.3 RCA selection criteria

Latino & Latino & Latino (2011, chapter 1) explain importance of RCA/EDA selection criteria in their book. The main question is how organization can balance development team's effort between RCA/EDA tasks and other activities. The number of issues and defects can be high, so the reality is that full analysis cannot be done for all of them. That is why an intelligent selection criterion shall be defined to ensure organization concentrates in right and valuable topics. (Latino & Latino & Latino 2011, chapter 1.) Linders (2014) state that high volume of executed analysis shall not be organization's target as such. It will just lead too many improvement action proposals, which the organization cannot deal with. Investing in analysing topics which will not finally implement any improvements, is waste and additional overload for organization (Linders 2014).

To stabilise the resourcing balance, one possible RCA/EDA selection criteria can be the evaluation of loss caused by the defect. If the impact to business value is significant, RCA is justified. Also, if high probability exists for similar problem to occur and cause losses in the future, RCA with preventive actions shall be done. (Linders 2014.) One viable way is to use a basic risk matrix tool, aligned with product's special parameters. Risk management matrix usage can ensure the analysis concentrates in defects or events which deliver the greatest benefit. (Dunn 2016.)

Severity / Probability	Catastrophic	Critical	Marginal	Negligible
Frequent	HIGH	HIGH	SERIOUS	MEDIUM
Probable	HIGH	HIGH	SERIOUS	MEDIUM
Occasional	HIGH	SERIOUS	MEDIUM	LOW
Remote	SERIOUS	MEDIUM	MEDIUM	LOW
Improbable	MEDIUM	MEDIUM	MEDIUM	LOW

Modified from: https://www.assetivity.com.au

Figure 3. Example of risk matrix (Dunn 2016).

2.2.4 RCA in complex systems and the future

The evolution and increased technical complexity of systems challenge current RCA/EDA process which has the basic idea about the one root cause which can be avoided to re-occur in the future by implemented improvement actions. In complex system root cause cannot necessarily solve all interaction problems no more. There are unexpected complex patterns, which cannot be all avoided by fixing the root cause. The analysis should concentrate instead of pure root cause, to find patterns and how to manage those. (Boeckman 2017.) According to Weeks (2018) RCA was more applicable when waterfall software development methods were used. Product's delivery cycle was months, even years and system stayed stabile and unchanged for a long time. In the age of agile and continuous deliveries, states of work are under constant change. RCA cannot always provide solutions and improvement actions quickly enough.

When complex system has a failure, it can be a combination of properties which occur because of different system elements working together. It might not be a defect in any single element or component. Instead of concentrating in one root cause, identifying contributing factors make sense.

There can be multiple interconnected causes and fix all of them is not always possible, reasonable, or cost-efficient. To recover from the incident and avoid it in the future, contributing factors shall be ranked and decide which ones are those to concentrate in. (Code[ish] 2021.)

Nevertheless, future of RCA/EDA is seen still overall strong as benefits are clear. RCA/EDA is still a tool which will bring visibility to underlying issues, at any phase of software development. With proper analysis organization will get critical information about risks and costs which are caused by non-conformance. (Eclipse 2022; Cross 2022.) And it is always cheaper to correct faults at early phase of the software development life cycle (SDLC). According to IBM's Systems Sciences Institution estimation, the cost to fix a bug is highly depending on when and in which SDLC phase the bug is discovered. The later the bug is found and fixed, the higher is the cost (Figure 4). If a defect slips to the maintenance phase and customer detects it, it is 100 times as expensive to correct it than when found at software design phase. So, the early detection of fault is not only improving the quality of software but also bring cost savings to organization. (Dawson et all. 2010, 51; Gitential 2022.) Costs are not only restricted to the functional code, but also test code associated with it could need to be rebuilt. When each testing level can have own separate test code which is affected by the change, it means costs and impacts of late phase found faults and faults slipped to customers, increase exponentially. (Churchwell 2018.)

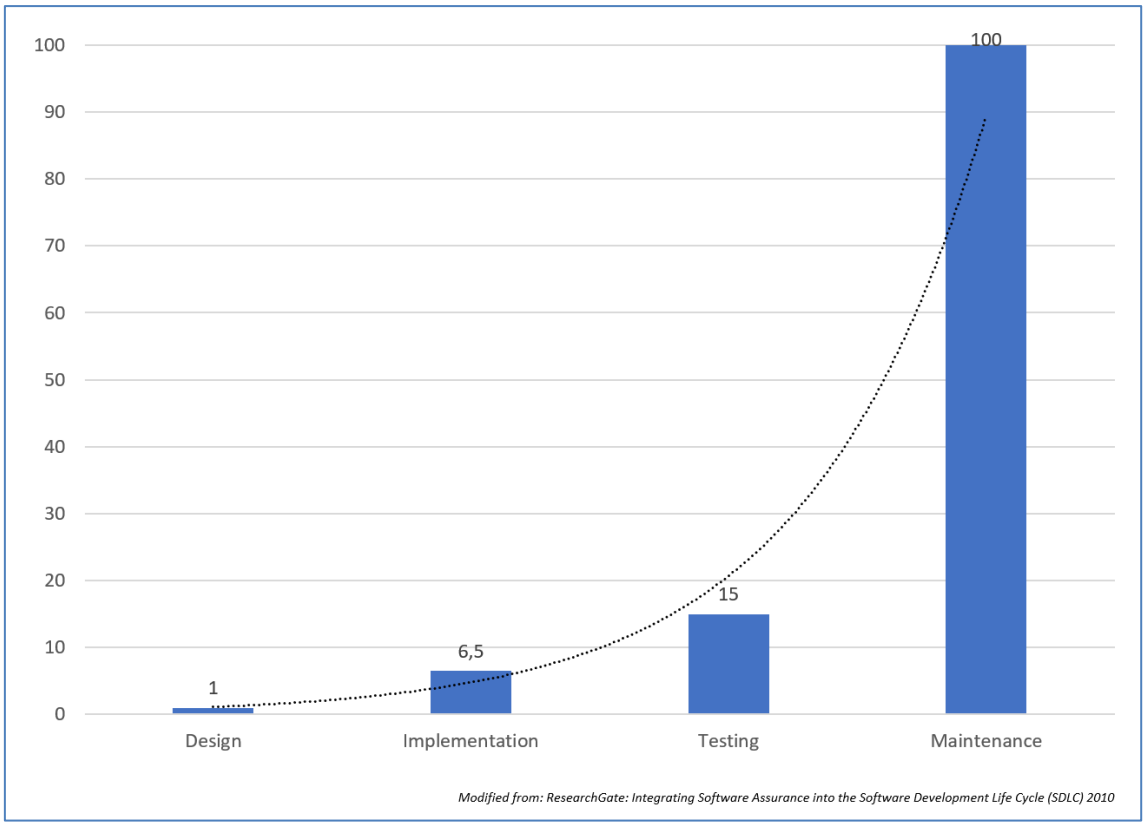


Figure 4. Relative cost of fixing software defects according to software development life cycle (Dawson et all. 2010, 51)

2.3 Continuous integration (CI) and continuous delivery (CD)

Modern software development practices are essential to fulfil customer expectations on applications in all technology areas. Main demands towards software are frequent software deliveries with fast update cycles and high quality. Continuous integration (CI) and continuous delivery (CD) is considered as a solution to meet those demands. CI/CD is defined as a collection of processes, defined practices, tools and common culture adapted in software development company. (Sacolick 2022.)

CI/CD starts from the change code commit to the version control repository done by software developer, ending up to the automatically prepared release to production (Figure 5). It is highly automated pipeline, which enables quick and reliable, incremental deliveries of changed code. Especially with agile software development practices the support of incremental work is in key role. CI/CD's aim is to reduce the risk of negative impacts to the whole project when parallel developers

and developing teams working simultaneously within the same product and in master/main software branch. The target of CI/CD is to decrease number of defects and conflicts during the integration of the complete project and ensure software changes are always deployable for customers whenever requested. (Sacolick 2022; GeeksforGeaks 2020.)

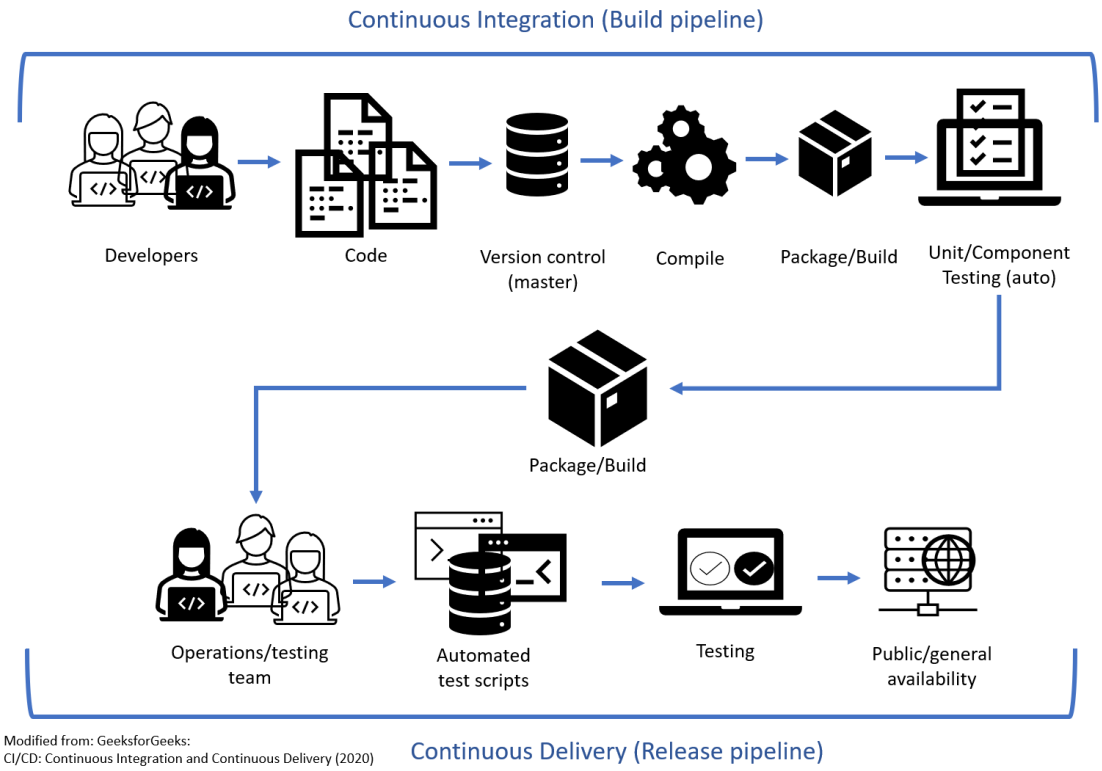


Figure 5. Continuous integration and continuous delivery process (GeeksforGeeks 2020).

2.3.1 Continuous integration (CI)

Continuous integration (CI) concentrates on the initial stages of software developed lifecycle. It covers steps from the actual piece of code commit made by the developer to the end of the initial testing. (Bigelow 2021 b.)

CI is explained by Sacolick (2022) as a toolkit of coding philosophy and practices which let product development teams to implement their code changes more frequently in smaller pieces to the version control repository. CI ensures all team members can integrate their work at least once per day to the master software branch with the help of automation. Automated build checks are done for

every integration to catch possible errors in master branch caused by those latest changes. The feedback loop is short, as after each commit software is built and tested immediately. (Sacolick 2022.)

This method is essential when software updates to customer need to be done in short cycles. Earlier, developers worked in isolated software branches during their feature or functionality development time and merged their changed code to master branch only after their work was completed. That kind of process was not only difficult and time consuming, but also very error-prone and led typically into unexpected breakages with long correction times. Keeping master branch always available, usable and stabile was hard. This is an area where CI/CD brings improvement and is now considered a core practice in the agile software development toolkit. (Amazon Web Services a.)

The key benefits of CI defined by Roy (2018) are improved developer's productivity, quicker finding and addressing defects and faster deliveries to customers. In practice, the productivity increases when developers are freed from executing manual task. As CI also helps to reduce number of bugs, it will improve overall cost efficiency. The shortened feedback loop improves the transparency of the entire process. It is easier for development team to determine the root cause of each failure when integration is done in short cycles, with a smaller number of simultaneous changes. (Roy 2018; Amazon Web Services a.)

2.3.2 Continuous delivery (CD)

Continuous delivery (CD) continues from the point where CI ends. CD's focus area is in the later phase of software development. It mainly covers the software build throughout testing, validating, and preparation for deployment. CD can also further deploy the software build but that is not a mandatory step in continuous delivery phase. (Bigelow 2021 b.)

CD relies on tools and automation. Those are used to lead a software build through advanced testing. Typical testing included in CD are functional, user acceptance, configuration, capacity, and performance testing phases. Key target for CD is to automatically validate that each build in hands meets customer requirements and can be used at production environment. Similarly, to CI also in

CD phase changes compared to earlier working build are rather small. It ensures potential problems identified during testing can be managed fast and less expensively than when using traditional software development methods. (Bigelow 2021 b.)

The target of CD is getting all kind of changes to production without breaking earlier working functionality. Changes can include for example configuration changes, new features, and bug fixes. CD makes deployment predictable and easier to schedule. The code shall be always in the state it can be deployed into further phases whenever required, no matter even software in master branch is under constant changes by developers. (Roy 2018.)

Benefits of CD are higher quality applications, reduced costs, and decreased risks for successful deployment. Since software release process is automated, it is efficient and rapid. While testing is already done on higher level, defects in software can be addressed fast, as number of changed parts is kept low. Also, more advanced testing can be performed because of the automated process. It will help to improve software quality. (Roy 2018; Amazon Web Services b.)

3 DATA COLLECTION

Data collection methods used in thesis are listed in this chapter. The chapter contains description of used datasources, data collection methods and tools. Also, data formatting, validating, modelling and visualization principles are described.

Companies can use both internal and external data in analysis and decision making. Internal data is considered to offer the inside look into company's practices and their effectiveness. Internal data will help to make improvement suggestions and target possible actions more accurate. As the data comes directly from company's own systems, it contains facts and information which are specific for the company in question. Internal data is not typically accessible by outsiders. (Ventiv Technology 2022.)

Data analysis in this thesis was fully based on the case company's internal data sources. Quantitative data was collected from company's inhouse tools and databases, qualitative data from interviews, organization's internal workshops and meetings, without a pre-defined questionnaire pattern. Since the main goal of this thesis was to find the pain points of the company's current RCA/EDA process and how to move RCA/EDA to an earlier stage of the software development life cycle, the use of internal data was seen as the best alternative. Open data that would have supplied enough detailed information to achieve the goals set for this thesis was not available.

3.1 Data sources and data collection process

In quantitative data analysis company's in-house software fault management, RCA/EDA and feature management databases were used as source. The actual data collection was made as a JavaScript Object Notation (JSON) query from Data Portal tool (Figure 6). Data Portal is an inhouse tool, a centralized online interface to the most important software development tools and databases. Data received from Data Portal tool cannot be considered as pure raw data, as on top of gathering data from various sources, it also makes some data formatting, like combining data from several fields into one column, executing calculations and storing calculated values.

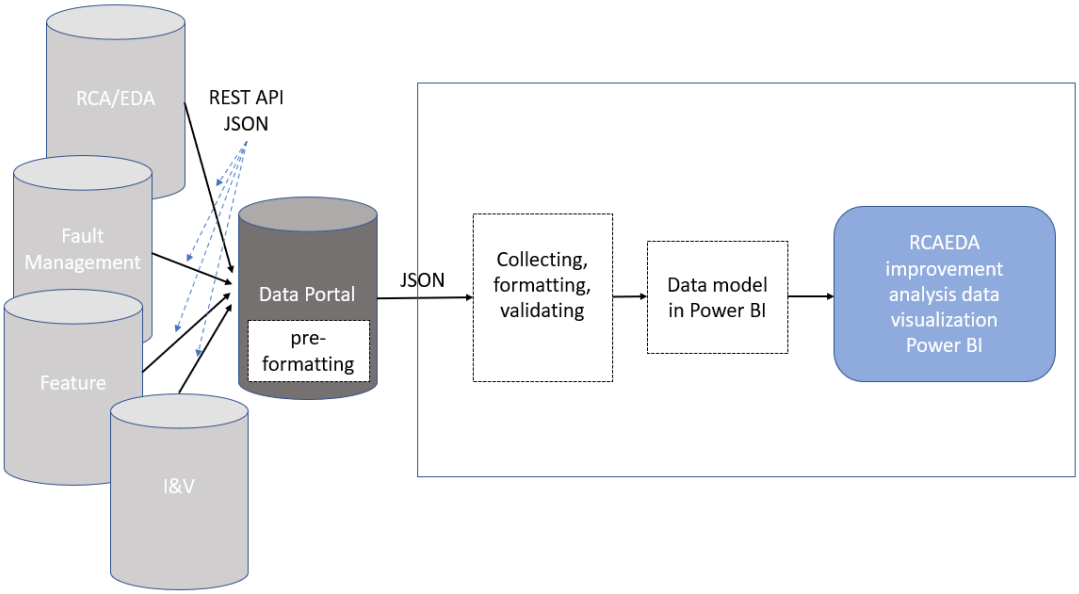


Figure 6. Quantitative data collection process

Microsoft's Power Query tool was used to collect data from Data Portal for analysis. The dataset was big, approximately 240 000 rows loaded from Data Portal by using three separate JSON queries. On top of that, four additional queries were implemented into Power Query. These additional queries were done by merging some data from original JSON queries.

Table 2. Power Queries and data source

Query name	Description	Data source
RCAEDAs	RCA/EDA data, all tickets created in predefined timeframe	RCA/EDA backlog
FaultReports	Fault report data, all tickets created in predefined timeframe	Fault management tool and I&V database, data combined by Data Portal
Features	Feature data, all features implemented in predefined timeframe	Feature backlog

RCAEDA_temp	Temporary query to combine each RCA/EDA ticket with fault report(s)	Modified from query "RCAEDAs"
RCAEDA_FaultReports	All RCAEDA data connected with fault reports	Nested join from queries "RCAEDAs" and "RCAEDA_temp"
Date table	Time stamps and calculated ages of fault reports and RCA/EDA tickets	Nested join from queries "RCAEDA_FaultReports" and "FaultReports"
Features_FaultReports	Combine features with fault reports, contains also calculated columns	Modified from query "Features"

After data collection, data validation was done in Microsoft Power BI's Power Query, Desktop's Data view and Excel. Data points, which differed significantly from other observations, were identified by using boxplots, data sorting and histograms. Sorting data to typical, unusual, and impossible values was done to help managing outliers. All unusual and impossible values were analyzed separately to decide whether they belonged in the data set and process as normal observations or whether they should be removed. Example of unusual value, which varied a lot from the median and did not meet the commonly used outlier detection rule of 1.5 x interquartile range (IQR), was a high RCA/EDA completion time (RCA/EDA age). After analysis, value was considered as valid data point instead of an outlier. Certain RCA/EDA tasks just took significantly longer to complete than others, so they were not errors and were kept in the dataset. There were only individual outliers seen, which were clear mistakes and not a normal variation of process execution. Those datapoints were removed from the dataset as correcting them was not considered as possible.

Interviews were made between October – November 2022. In total four persons were interviewed. Interviews were private, one person at a time and were arranged via virtual conference. Participants were selected based on their roles. Two participants from RCA/EDA practical management and two participants from execution were participating in voluntarily interview. There was no formal transcript shared with participants beforehand. Interview was a non-formal discussion about

company's current RCA/EDA practices and how participants saw the future of RCA/EDA when continuous integration and continuous development practices would become mainstream methods. Interviews were made in English, and they were not recorded. Notes were made manually during the interview without adding the names of the participants. More insights were gathered by attending two separate improvement workshops related to real-life use cases in the company's RCA/EDA work. All answers and insights are handled anonymously in this thesis, without possibility to connect answers to the exact person.

3.2 Data model and visualization

Data model was built up in Power BI / Desktop's Data view. Relational connections between data tables were built up. Most of the connections were made automatically by Power BI, but some of them needed manual addition. Data structure, including relations between RCA/EDA data and fault reports is defined in Appendix 2.

Data visualization was done in Power BI / Report view. On top of Power BI's default visuals, also additional visuals were used: R script visual, Correlation Plot, Pareto, and Box and Whisker. Microsoft Excel was used as a supportive tool to generate graphs and visuals which were able to be shared outside the company.

4 ANALYSIS RESULTS

The aim of this thesis was to identify problematic parts in company's current RCA/EDA practices and how to left shift the RCA/EDA work to earlier phase of the software's development cycle. Analysis concentrated primary on data of current RCA/EDAs and identifying patterns connected to RCA/EDA work. Secondly, correlation between different software development related parameters and RCA/EDAs was investigated. The final part of the analysis was a deep dive for defects reported by customers, as improving the current practice required detailed view on software defects slipped to customers so far.

As company's internal data was used as a base of analysis, it was not possible to show in this report all graphs and visualizations in original format because of confidentiality restrictions. So, all company confidential information was hidden from analysis results. This means for example graphs and other visuals were generated by using ratios and trends instead of actual volumes, efforts, and time related parameters.

The quantitative analysis concentrates in company's root cause related data, patterns and possible correlations between several factors connected to RCA/EDA tasks. Summary of quantitative data-analysis main topics are listed in table 3. Qualitative data analysis target is to offer more insights related to issues handled in quantitative analysis. Analysis results presented in this chapter are combined based quantitative and qualitative data.

Table 3. Summary of data analysis topics in thesis.

Name	Description	Data source
RCA/EDA analysis selection criteria and coverage	Created RCA/EDA request based on the TL9000 problem report severity classification and defect originator	RCA/EDA and fault management databases
Volume of RCA/EDA requests and tasks	How many fault reports contain RCA/EDA tasks, divided into distinct categories	RCA/EDA and fault management databases, interviews

RCA/EDA's lifecycle	What is the lifecycle of RCA/EDA in case company and main factors impacting to the RCA/EDA age.	RCA/EDA and fault management databases
RCA/EDA team setup	How the RCA/EDA work is done inside teams, which roles are in charge.	Interviews
Improvement action completion ratio	Completion ratio of improvement actions divided to distinct categories: software area, root cause and TL9000 problem severity classification.	RCA/EDA and fault management databases, interviews
Impact to customer	RCA/EDA execution based on different impact categories.	RCA/EDA and fault management databases

4.1 RCA/EDA selection criteria and coverage

First, RCA/EDA selection criteria was examined. Company's process defined that all software defects requiring RCA/EDA from software development and testing areas, should have been selected either by following TL9000 problem severity classification (see appendix 1) or special internal rules.

Figure 7 presents coverage of created RCA/EDA tasks per fault reports and problem severity classes. Values were graphed based on fault report's originator (customer / internal). In case of customer identified defects, all severity class "Critical" fault reports had RCA/EDA task created, so the coverage was full 100%. From severity class "Major" 93% of cases got RCA/EDA request. When analysing company's internally reported defects, there was not seen that clear connection to TL9000 classification, but more to company's internal rules defined in RCA/EDA process. Every third internal fault report with severity grading "Critical" got the RCA/EDA request and from the group of severity "Major" RCA/EDA request coverage was 13%.

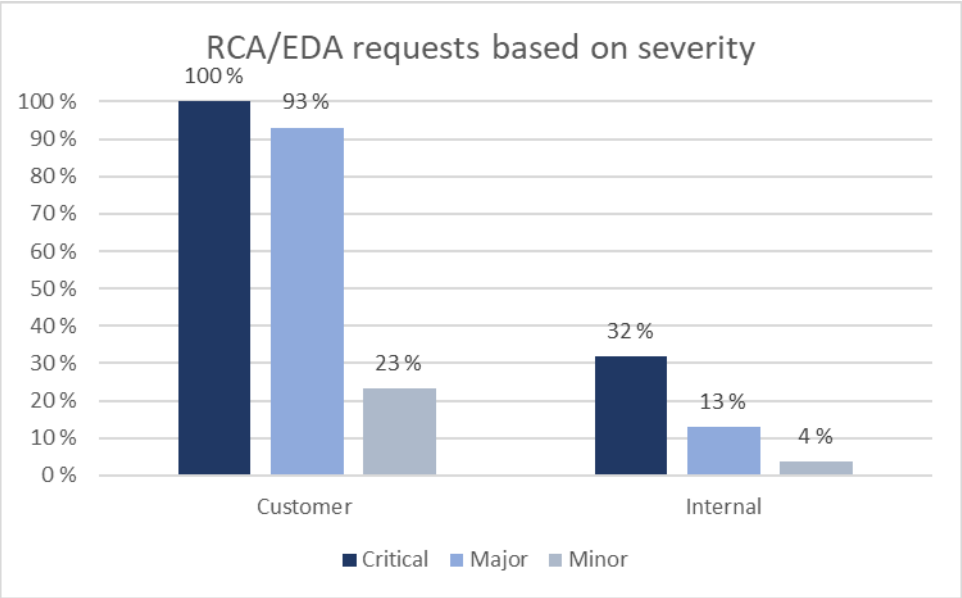


Figure 7. Created RCA/EDA tasks based on the TL9000 problem report severity classification and defect's reporter

4.2 Volume of RCA/EDA requests and tasks

Next, volume of RCA/EDA requests and tasks was investigated. The relational connection between fault reports and RCA/EDA tasks was many to many. It means more than one RCA/EDA task could have been attached to an individual fault report and vice versa (see appendix 2).

Figure 8 visualizes the proportion of fault reports that contain an RCA/EDA request. In total, RCA/EDA was requested to 17% of all soft-ware fault reports created during the observation period. When all defect reports with an RCA/EDA request were reviewed according to the origin of the defect, 32% were reported by the customer and 68% were reported internally. On the other hand, when all individual RCA/EDA tasks were reviewed at the level of their associated defect report originator, the picture was different: 74% of RCA/EDA tasks were related to customer-detected defects and only 26% of RCA/EDA tasks were related to internal company defect reports. So, in this case, the internal bug reports contained 2/3 of the RCA/EDA requests, but when the final RCA/EDA related tasks (analysis and improvements) were created in the RCA/EDA tool, 3/4 of the tasks related to customer identified bugs. This means that the analyses of defects found internally contained fewer sub-tasks than the analyses of defects from customers.

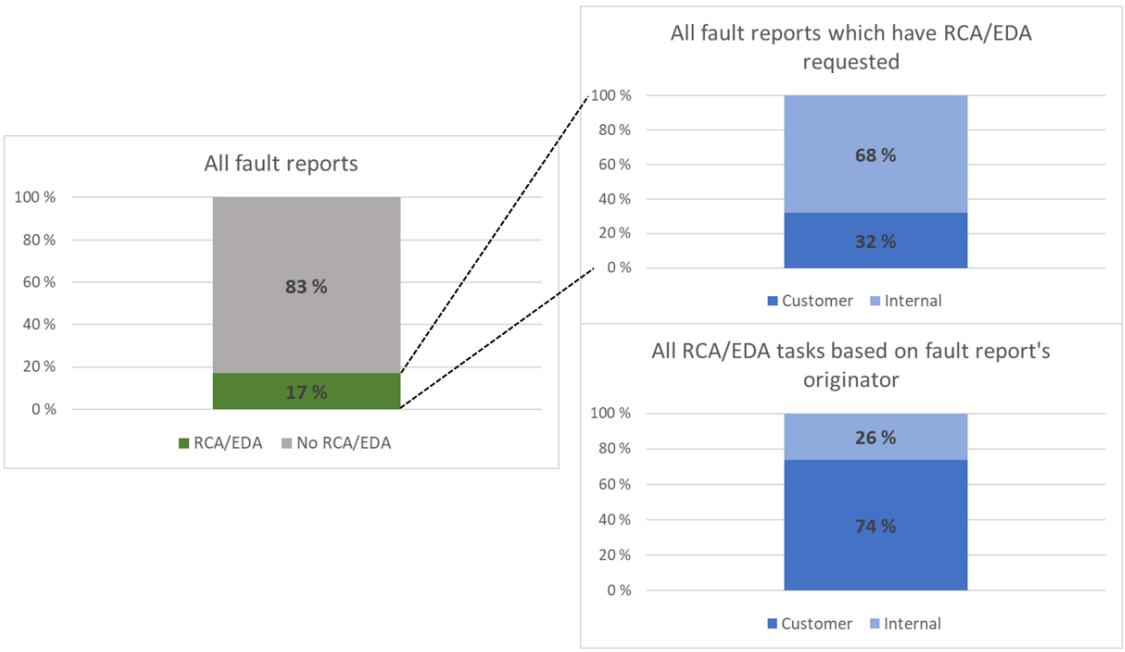


Figure 8. All fault reports and RCA/EDA tasks based on the defect's reporter

More insights from RCA/EDA tasks were received by using Pareto principle. The idea was to get a view how RCA/EDA tasks were divided between different software development areas. It was noticed that 80% of analysis were allocated towards six software development areas. Inside that group, there was one software area holding clearly higher number of analysis tasks than others (Figure 9).

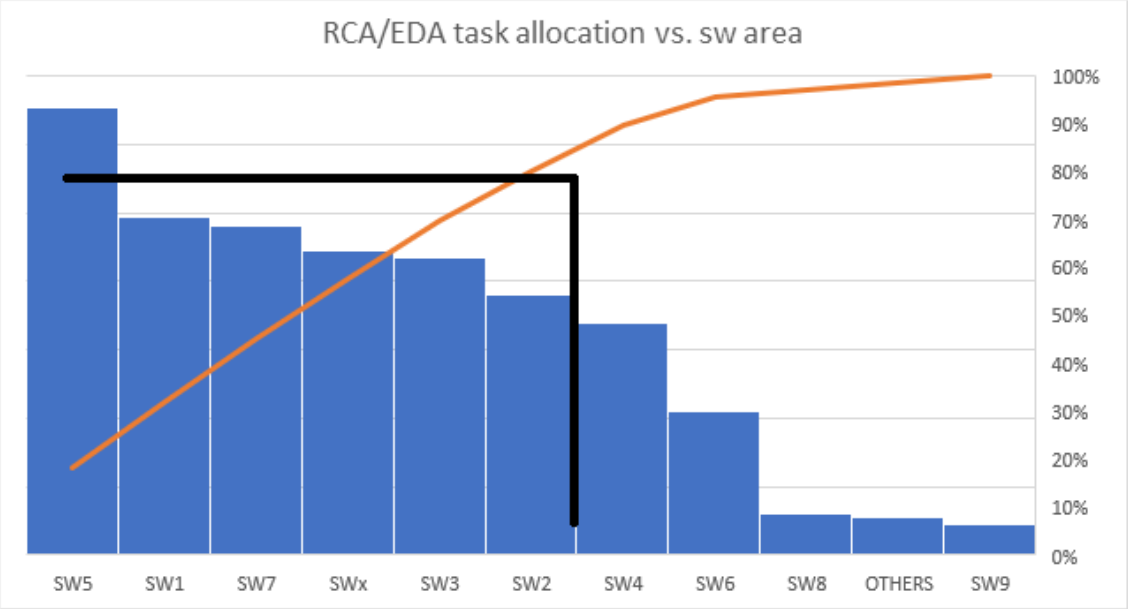


Figure 9. RCA/EDA main task allocation per software development area

In interviews the volume of RCA/EDAs was discussed, and participants shared the common view about requests. It was considered as high, and especially number of RCA/EDA requests towards internal fault reports.

4.3 RCA/EDA's lifecycle

As a first step in lifecycle analysis the RCA/EDA's age was compared to fault report's age to detect possible differences or different patterns. The age of RCA/EDA task was calculated from task's creation to its closure.

The average (mean) RCA/EDA age during the observation time was 2,4 times higher than average fault report's age during that same period. In other words, if average fault report's closure time would have been for example 5 days, RCA/EDA task would have taking 12 days to complete. When checking median (middle value), the difference between fault report's age and RCA/EDA's age was smaller, but still clearly visible. The median age of RCA/EDA was 1.6 times greater than the median of the fault report (Figure 10).

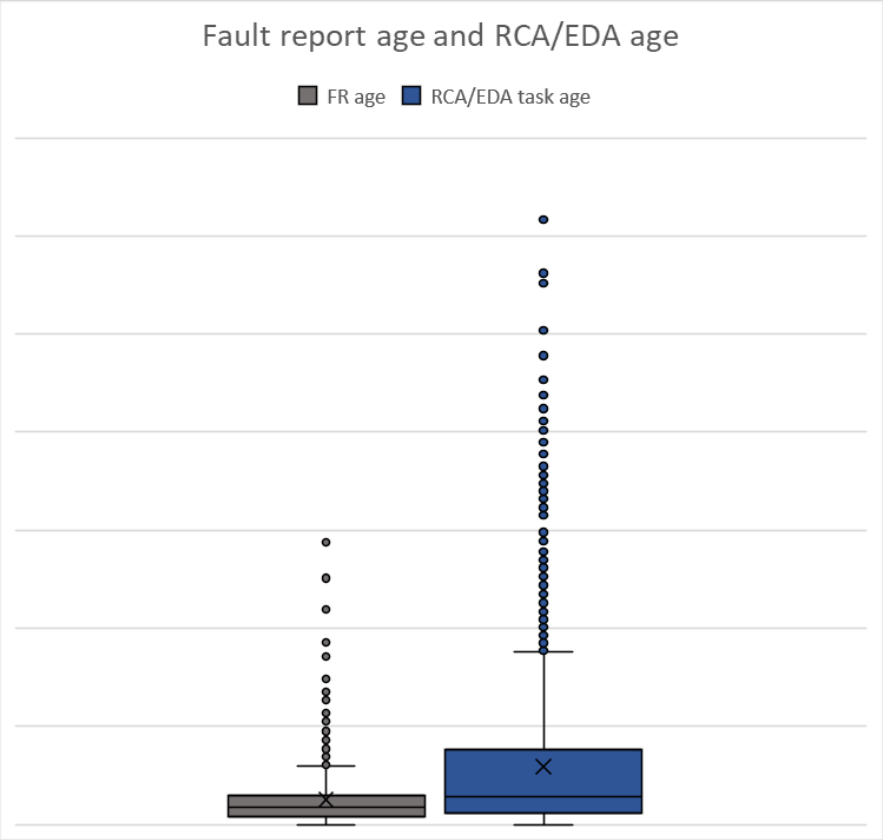


Figure 10. Fault report's age compared to RCA/EDA's age (Box and Whisker Plot)

Nevertheless, there was no monotonic relationship seen between RCA/EDA's age and fault report's age. If closing a fault report took longer, there was no visible statistical correlation to the analysis age. Also, there was no dependency on RCA/EDA's age with the timing of actual analysis task creation. Thus, it had no effect on the age of the RCA/EDA, how quickly after the bug report creation or the bug report closure, the analysis task was generated into the tool (Table 4).

Table 4. Kendall's rank correlation of some RCA/EDA related timestamps. More information in Appendix 3.

	FR age	*RCA task created after FR creation*	*RCA task created after FR closure*	*RCAEDA age*
FR age	1			
RCA task created after FR creation	0,37	1		
RCA task created after FR closure	-0,13	0,52	1	
RCAEDA age	**0,01**	**-0,01**	**-0,03**	1

Second step in RCA/EDA age analysis was to compare analysis age on fault report's originator level, between customer and internally reported defects.

The analysis of RCA/EDA completion age pointed out some level of difference between originators. Overall, average completion time of customer originated RCA/EDAs was 1,3 times shorter compared to age of analysis for internally originated defects. In median there was no difference (Figure 11).

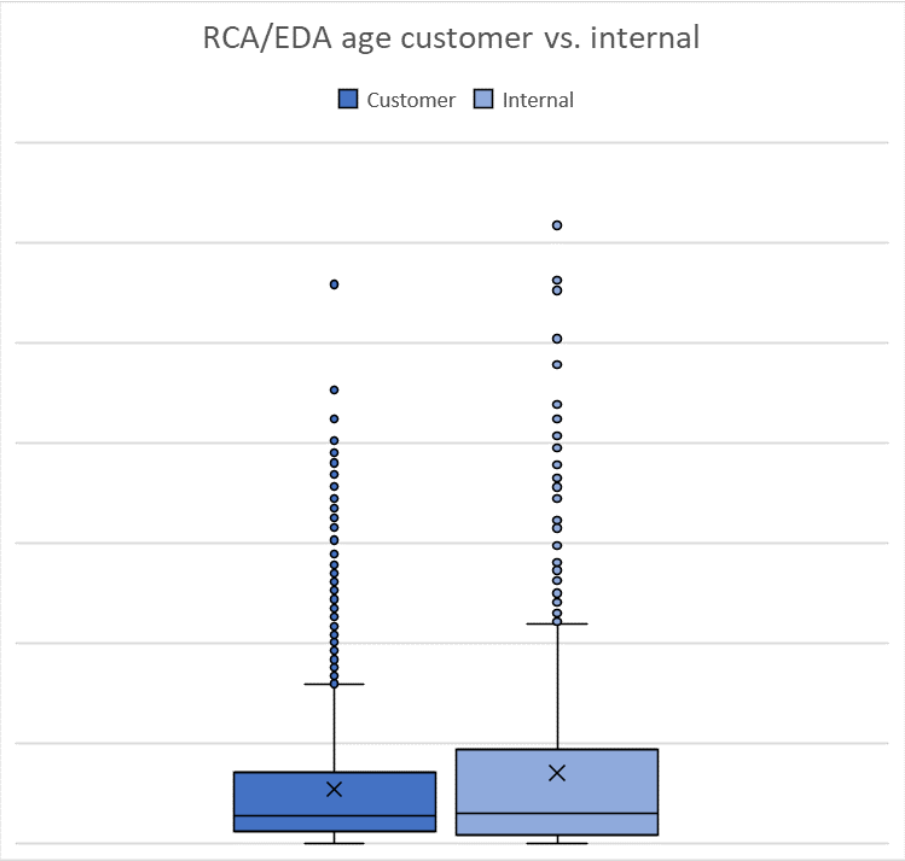


Figure 11. Comparing RCA/EDA age between customer originated and internally reported fault reports (Box and Whisker Plot)

4.4 RCA/EDA team setup

In interview there was discussion with participants how the RCA/EDA team was set up in their software areas. There were clear roles defined in process to handle the execution of RCA/EDA. This was followed by all organizations. Each organization had for example persons nominated to handle those RCA/EDA roles. But at the end, also the actual software designer, who had the technical expertise on that area and particular case, was typically invited into RCA/EDA work.

4.5 Improvement action completion ratio on software area level

Each RCA/EDA 5Why analysis request contained several sub tasks in database. Under main RCA/EDA tasks there were own sub tasks for detailed analysis and improvement actions. Structure is explained in Appendix 2.

Next, those improvement action tasks were taking under investigation. The first analysis was to check how many of those fault reports in which RCA/EDA was asked for, the analysis was completed, but no improvement measures were identified. According to statistics more than half (54%) of fault reports where analysis was completed, were missing all improvement actions. When these cases without improvement actions were investigated on software area level, it was noticed that 33% of those where improvement actions missed, were for one software area (Figure 12).

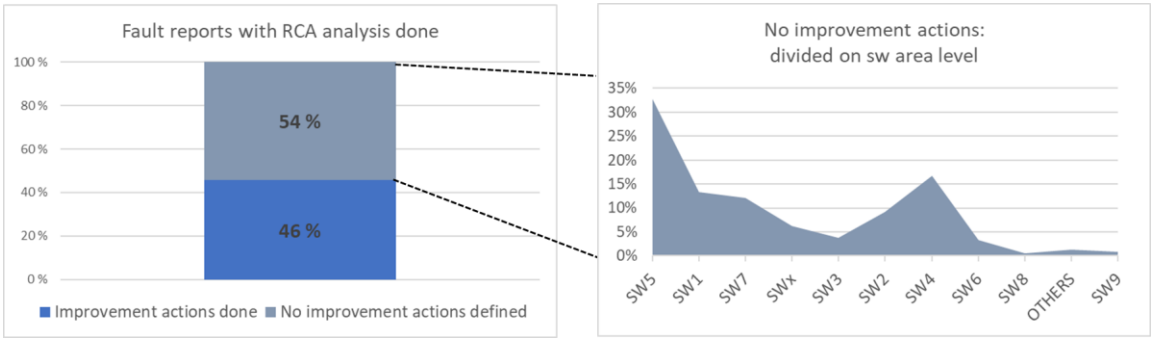


Figure 12. Fault reports where analysis done, but no improvement actions identified or implemented.

More detailed view on software area improvement action completion ratio was generated next. It showed significant difference between areas. Improvement action completion ratio varied from 25% to 75%. Six out of ten (6/10) software areas were able to identify and implement improvement actions to more than half of their RCA/EDAs whereas four out of ten (4/10) areas stayed below 50% improvement action completion ratio (Figure 13).

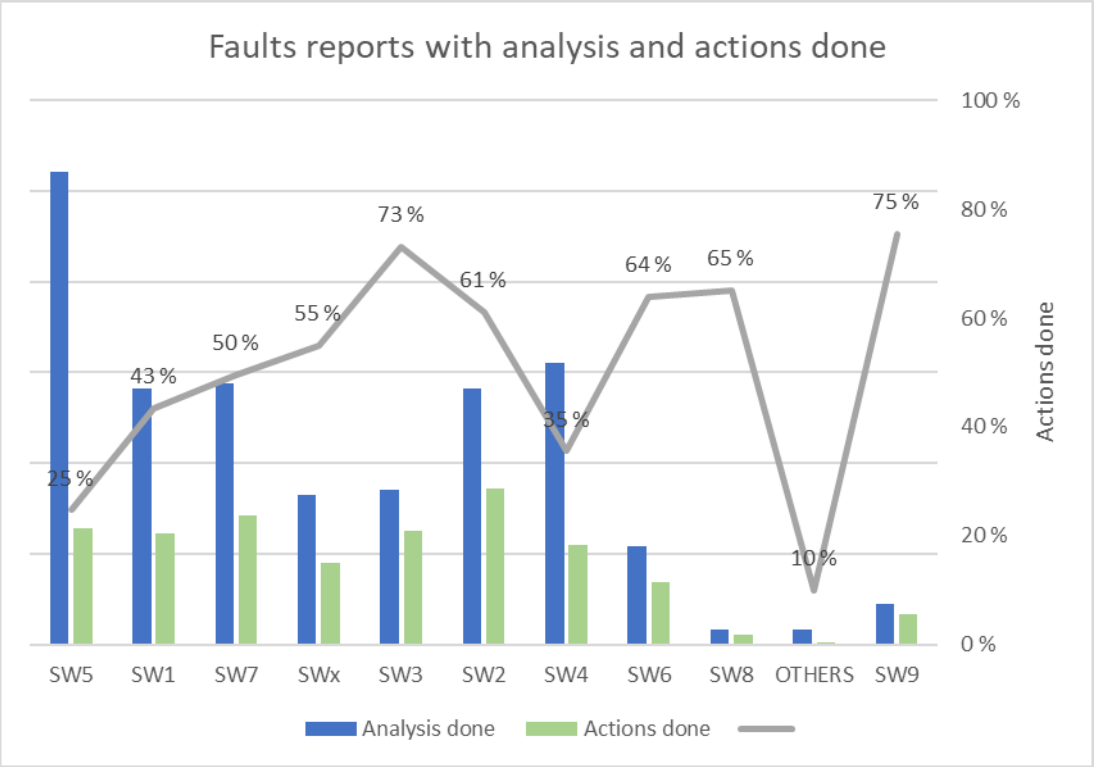


Figure 13. Detailed view on improvement action completion ratio per software development area

Insights gathered from interviews and workshops pointed out also the challenge in identifying proper improvement actions in certain areas and in all specific cases. It was considered impossible to skip the RCA/EDA, as executing the analysis is always mandatory to be executed for any case it is requested.

4.6 Improvement action completion ratio based on TL9000

After improvement action completion ratio check on software area level, identified and implemented preventive actions were investigated on fault report's TL9000 severity classification (Appendix 1) level. This was done by separating customer originated fault reports and internal fault reports to see if there was some variation depending on defect's originator.

There was significant difference in preventive action completion ratio based on reporter of the defect. While 80% of Critical severity customer defects, where RCA/EDA was requested and done also got preventive actions completed, for internal fault reports the rate in Critical severity class was 36%. In other words, only 1/3 of the analyses requested and performed resulted in any kind of

product improvements. The same trend was also seen in the severity category Major. Only in the Minor category was the completion rate of preventive actions the same for defects reported by the customer and internally reported ones (Figure 14).

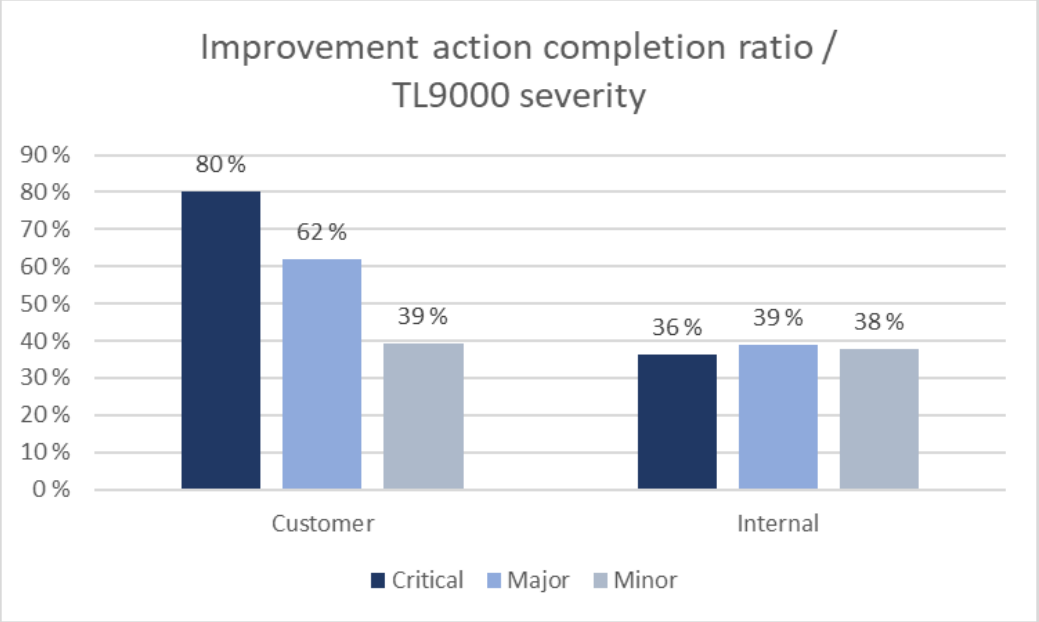


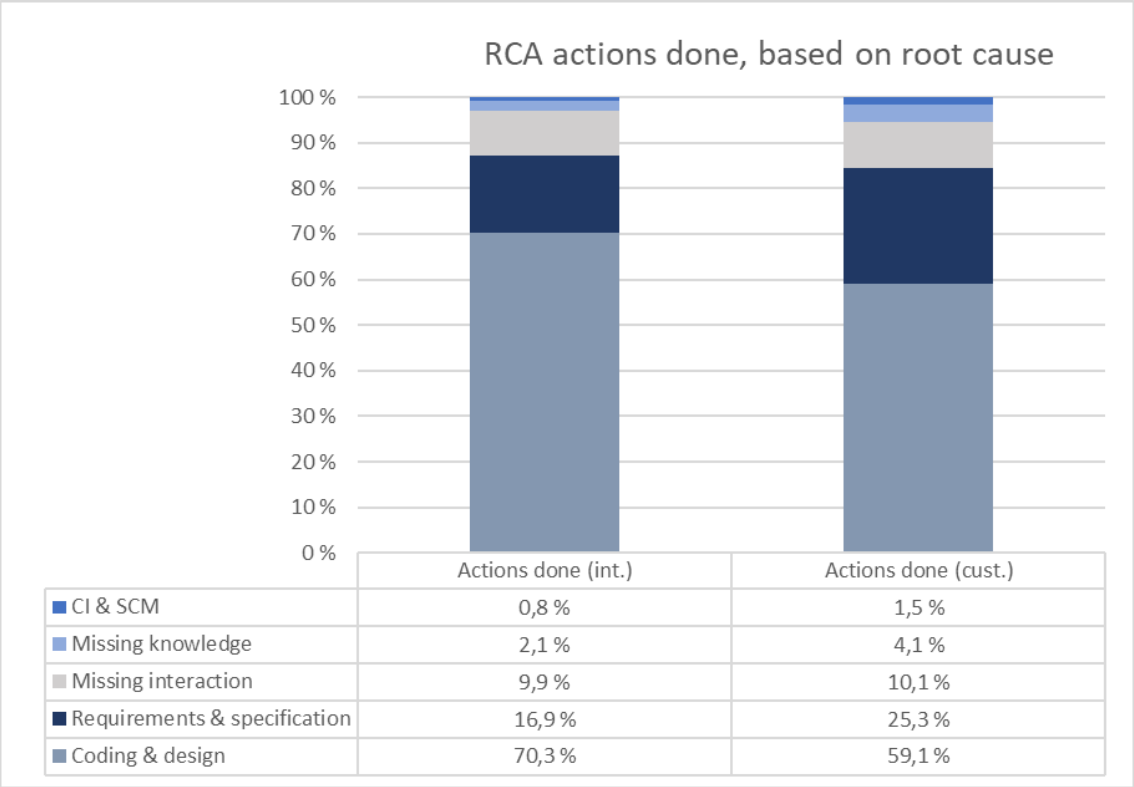
Figure 14. Detailed view on improvement action completion ratio per fault report's originator and TL9000 problem report severity classification (Appendix 1)

4.7 Improvement action completion ratio based on root cause category of the defect

Root cause of the defect and its connection to improvement actions was analysed at this point. First, root causes were divided into five distinct categories, based on root cause's area. Categories were:

- requirements and specifications
- coding and design
- continuous integration (CI) and software configuration management (SCM)
- interaction
- knowledge

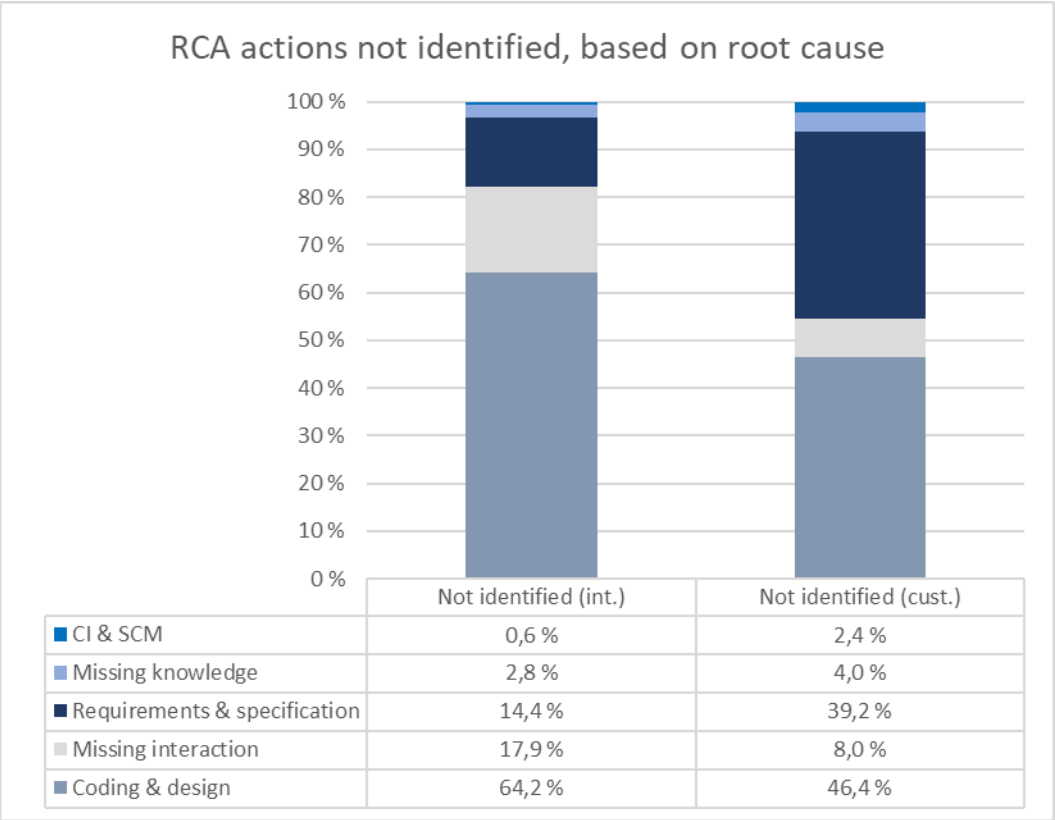
When analysed all improvement actions which were completed, the largest root cause category of fault reports was coding & design. The second biggest category was requirements & specifications. These two areas accounted for more than 80% of actions performed due to defects reported by the customer and almost 90% of procedures for internal defects (Figure 15).



	Actions done (int.)	Actions done (cust.)
CI & SCM	0,8 %	1,5 %
Missing knowledge	2,1 %	4,1 %
Missing interaction	9,9 %	10,1 %
Requirements & specification	16,9 %	25,3 %
Coding & design	70,3 %	59,1 %

Figure 15. Fully completed RCA/EDAs where improvements identified and implemented, based on root cause of fault reports

The other part of the preventive actions check was related to root cause of those fault reports, where RCA/EDA was done, but no actions identified or implemented. Both for customer and internal defects it was seen the highest root cause category with missed improvement actions was coding & design category. The difference between customer and internal fault reports com-pared to preventive actions was seen in second biggest category. For customer faults it was requirements & specifications category, but in case of internal faults, missing interaction was the second biggest category where no actions were done via RCA/EDA (Figure 16).



	Not identified (int.)	Not identified (cust.)
CI & SCM	0,6 %	2,4 %
Missing knowledge	2,8 %	4,0 %
Requirements & specification	14,4 %	39,2 %
Missing interaction	17,9 %	8,0 %
Coding & design	64,2 %	46,4 %

Figure 16. Fully completed RCA/EDAs where preventive actions were not identified or implemented, based on root cause of fault reports

The final check for the root causes was to analyse statuses of preventive actions per each root cause category of internal fault report. The idea was to check if there were differences in completed and non-completed actions based on root causes. And it also included a comparison with the number of customer fault reports received, so that it was possible to investigate from which root cause category the faults leaked to customers (Figure 17).

Category of coding and design was clearly the biggest one. Most preventive actions done and not done belonged under that category. As well highest number of received customer faults. But the biggest gap between improvement actions done compared to faults leaked to customers, was in category of requirements and specifications root cause area. When checked the total volume of requested RCA/EDAs (improvement actions done + not done) in that category, it was not reaching the level of fault reports received from customers. Overall, in each category number of not identified/not done preventive actions for completed RCA/EDAs was higher than implemented preventive actions.

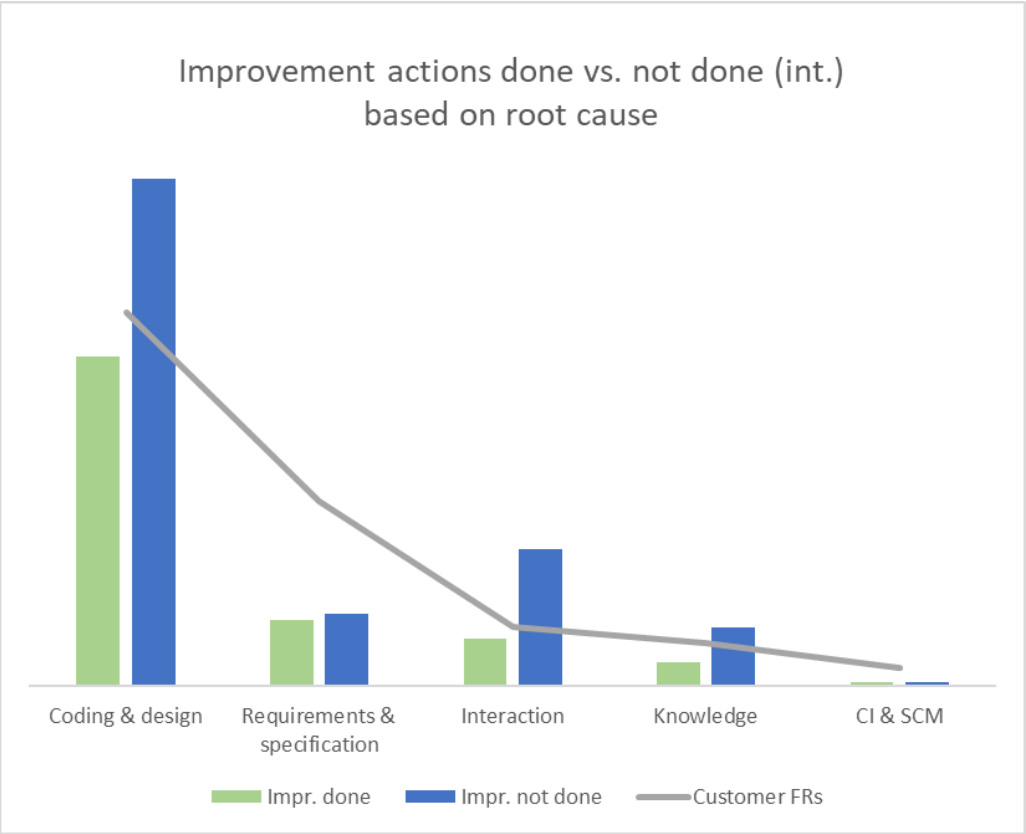


Figure 17. Improvement action status for internal fault reports based on defect's root cause. Grey line shows the number of customer fault reports received from each root cause category.

4.8 Impact to customer

Next, the fault report's customer impact connection to executed internal RCA/EDAs was checked. The idea was to gather information from which impact categories faults were leaking to customers and what was the balance between categories in RCA/EDA work. It was done by visualizing how internal fault reports with RCA/EDAs were completed in each category. The target was to get a view which is the level of completed RCAs and implemented preventive actions in those categories from where customer fault reports were received.

First, fault reports were divided into four severity categories according to example risk matrix (Dunn 2016). Categories in the priority order: Catastrophic, Critical, Marginal and Negligible.

Statistic showed in categories Critical and Marginal, that there were almost the same number of preventive actions completed. But volume of customer defects was almost duplicate in Critical category compared to Marginal. In Critical category there were more RCAs completed without improvement actions, than in any other category (Figure 18).

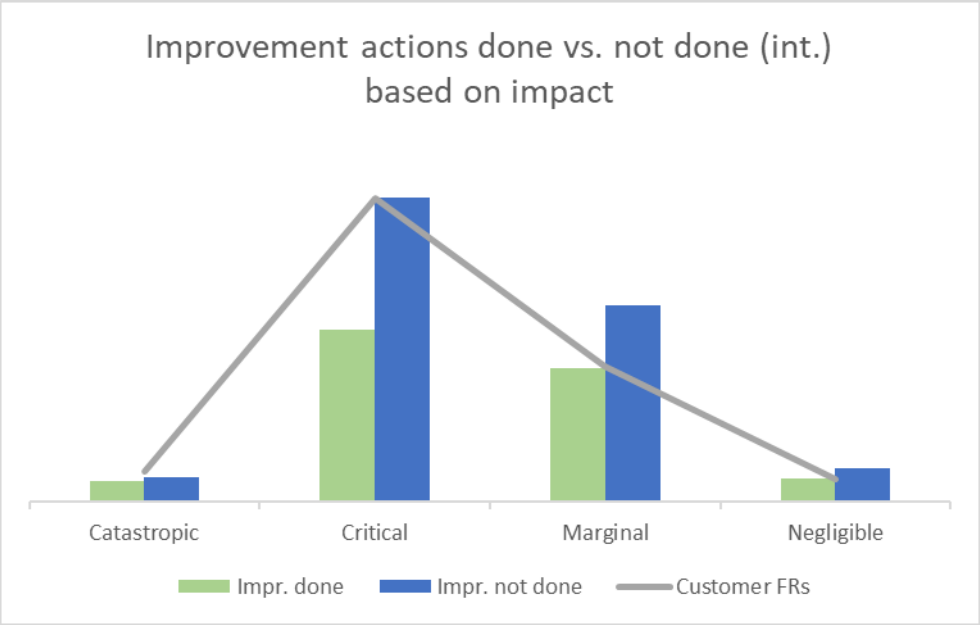


Figure 18. Improvement action status for internal fault reports based on defect's root cause. Grey line shows the number of customer fault reports received for each root cause.

Next, the volume and probability of customer defects were considered, and fault reports were divided into four classes based on their risk (Dunn 2016): RED, ORANGE, YELLOW and GREEN.

RED was the highest importance class, and it contained the biggest number of customer defects, completed and non-completed improvement actions. Number of customer defects in other classes was clearly lower when compared to RED class. In each risk class number of RCAs done without any preventive actions identified was higher compared to improvement actions completed (Figure 19).

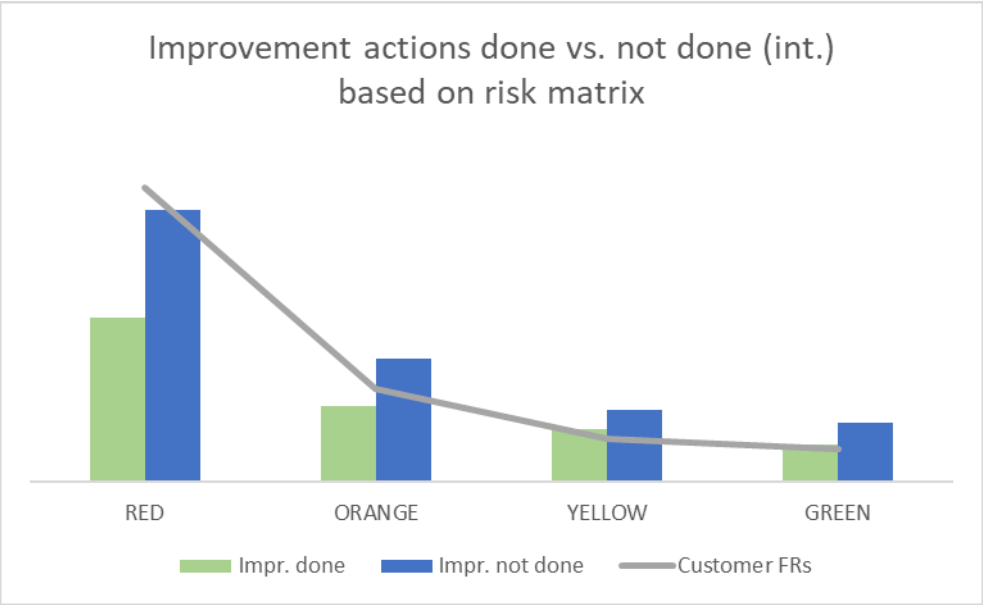


Figure 19. Improvement action status for internal fault reports based on the defect's risk level (Dunn 2016). Grey line shows the number of customer fault reports received for each root cause.

5 FINDINGS AND DISCUSSION

This chapter presents the results of the study in detail. Each research question is reviewed separately, interpretated findings are listed and discussed based on the background studies. The chapter is structured according to key topics.

Pain points in company's current RCA/EDA practices are discussed first. Then improvement ideas based on data-analysis results and identified pain points are listed. The last topic contains a proposal how RCA/EDA work can be left-shifted to the earlier phase in software development's lifecycle. At the end of this chapter the research process and known limitations of the analysis are presented, as well recommendations for future study are shared.

5.1 Main pain points in case company's RCA/EDA practices

There are three main pain points found in company's current RCA/EDA process and practices. Main problem noticed is related to misbalance between named preventive actions and completed root cause analysis. Especially what comes to company's internal fault reports, there is high volume of analysis done in certain areas without any actual improvements completed when at the same time defects from those areas are leaking to customers. Second problem is related to RCA/EDA tasks lifecycle. Finalizing RCA/EDAs take too long, so they should be completed faster to support CI/CD and agile development practices better. The third problem is the high volume of requested RCA/EDAs for internal fault reports. Rules for selecting fault reports under RCA/EDA are too simple, and not taking the customer impact and benefit to business enough into consideration. Even though improvement actions are done, the impact of actions is not fully seen in customer fault inflow.

5.1.1 RCA/EDAs with no preventive actions

High volume of RCA/EDAs where real improvement actions to software product are missing is seen as the biggest problem based on the analysis. As stated by Latino & Latino & Latino (2011, chapter 1.) and Linders (2014), the target shall never be only the high volume of RCA/EDAs done, but instead to concentrate in topics which bring real value to the company and to its customers. Initial

target of RCA/EDA is to find solutions which avoid the same problem to happen again, so RCA/EDAs without any improvements identified can be considered as waste in agile software development. (Linders 2014.)

Results show that over 50% of executed root cause analysis ended up with no improvement actions done to the product under development (Figure 12). Every third (34%) of those analysis without improvements identified, were towards one single software area (Figure 13). In details, that area had only 1/4 of RCA/EDAs executed with improvements and 3/4 finalized without any identified preventive actions.

When comparing RCA/EDA improvement action completion ratio between internal and customer originated fault reports, the clear difference was seen (Figure 14). For internally originated fault reports, all completed analysis led to improvement actions only in 36-39% of cases, depending on fault report's severity class. But for customer cases, the improvement action completion ratio was 80% for highest severity (Critical) and 39-62% for other severities (Major and Minor). This means more attention to RCA/EDAs of internal cases is needed to gain the real benefit out from those analysis. For customer originated fault reports RCA/EDA improvement action coverage ratio is already on the level considered bringing value to the company.

5.1.2 Long lead time of RCA/EDA

The life cycle of RCA/EDA is currently too long to fully support the short delivery cycles of CI/CD. According to Weeks (2018) especially when agile software development methods with CI/CD are in use, it is important to execute RCA/EDAs fast after solving the problem as states of work are under constant change. Also, lessons learnt shall be always a part of the RCA/EDA practices to ensure analysis of possible other improvement opportunities into product or process. Lessons learned must be recorded and shared within the organization to avoid added duplication of work. (Barsalou 2014, 45; Etti-Williams 2017.)

In data analysis, the RCA/EDA age for internally originated fault reports was 1.3 times higher than for customer reported defects (Figure 11). This difference can be explained by priorization. Customer originated cases are considered to reach higher priority compared to internal cases. Until certain level this is understandable way of work, but when thinking CI/CD practices, shorter lead

time of internal RCA/EDAs with preventive actions is essential and bring value. The earlier the fault is found and corrected, the cheaper the fix is for the company. When fault slips to customer, it is already one hundred times expensive to correct than in earlier software design phase. (Dawson et all. 2010, 51; Gitential 2022.)

For internally reported defects, the long lead time of RCA/EDAs means that possible improvement actions won't take place in fast enough to reach next customer software deliveries. In can lead to new unnecessary incoming customer defects which would have been avoided by faster finalization of the RCA/EDA and the shared lessons learnt. Also, long feedback loop is problematic from agile software development practices point of view. It is always considered an interruption of agile work, when the developer shall return to an old implementation that was completed a long time ago. According to Kim et al. (2016, part 1, chapter 2) agile software development's context does not allow waste, like multitasking and task switching with different uncompleted tasks. RCA/EDAs can cause unforeseen impacts to new feature development as same developers can be already allocated in the new feature development which is disturbed by additional RCA/EDA requests. (Bernstein.)

Too long a life cycle can also lead to an analysis where improvement actions are no longer possible or reasonable because the design of the software in that particular area has already changed completely after the fault was identified and corrected. It is also seen that coding and design is the biggest root cause category where finalized analysis led to non-identified and non-implemented preventive actions (Figures 15-17). This is also waste in agile software development and cause unnecessary costs to company.

5.1.3 Number of internal RCA/EDAs

The volume of RCA/EDAs requested for internal fault reports is seen as a problem. Latino & Latino & Latino (2011, chapter 1) explain in their book the importance to control the number of RCA/EDA requests. Organization shall be able to balance development team's effort between RCA/EDA tasks and other activities. If there are high number of issues, it is unrealistic to expect full analysis for all of them. Linders (2014) also point that high amount of analysis lead into improvement action proposal overload, which organization cannot anymore manage. As defined by Tableau Software (2022), the main driver to execute RCA/EDAs is the overall improvement in efficiency. It is better

to have systematic, planned preventive actions than just handle ad hoc symptoms, work in fire-extinguishing and reactive mode. (Tableau 2022.) That is why it is important to pay attention to RCA/EDA requests and their selection rules.

According to data collected during the research period, for customer cases the rules based on TL9000 problem severity classification (Appendix 1) seems to be working well, but for faults reported internally rules are not that straight forward. When for customer cases RCA/EDA was requested to highest severity categories with more than 90% coverage, for internally originated fault reports 32% of severity class Critical and 13% of category Major got RCA/EDA requested. So, the selection criteria for internal cases are not guided by TL9000 standard definition but they are done according to company's own, internally defined rules. Those rules are based mainly in impact related to company's R&D. Rules is typically fixed, so all internal cases meeting the criteria, will get RCA/EDA requests by default. Requests are typically created to fault reports currently under handling.

In figure 8 it is seen that from all requested RCA/EDAs, two-thirds are towards internal fault reports and only one-third against customer originated cases. But despite the higher number and proportion of requests, the preventive actions taken regarding internal fault reports have not prevented the inflow of incoming customer fault reports. In figures 17, 18 and 19 internal fault reports and their RCA/EDA requests (completed and not identified improvement actions) are compared with number of customer defects received during the observation time. There are clear gaps in some areas, for example low volume of RCA/EDAs requested in requirement and specifications root cause area compared to incoming customer defects towards the area (Figure 17). Another observation is that when customer impact of internal fault report is on marginal level, there are still big volume of RCA/EDAs requested and improvement actions are completed whereas the inflow of new customer cases is not that high there compared to more severe impact categories (Figure 18).

5.2 How can RCA/EDA better support continuous deliveries?

Based on the pain points listed in previous chapter and analysis results, there are four topics seen as major opportunities for improvement areas in RCA/EDA to better support continuous deliveries.

Concentrate in internal fault reports. The main finding based on data-analysis is, that concentrating in internal fault reports and their RCA/EDA practices has the best opportunity for improvements. As volume of internal fault reports and RCA/EDA requests for internally originated fault reports is high, also improvements done on that process area will have the biggest impact. Customer reported faults can be used as a reference for improving internal RCA/EDA practices, but overall customer fault related RCA/EDAs are already handled more efficient compared to internal ones.

Decrease the lead time. First action to improve RCA/EDA process and practices is to decrease the lead time of RCA/EDA. It is needed, although there is a risk of potentially weaker RCAs/EDAs if these actions are emphasized too strongly. The problem with lead time looks to be more like a problem with RCA/EDA request volumes, priority, targeting and resourcing rather than any individual RCA/EDA process factor as such. There was for example no statistical correlation between RCA/EDA's age and the possible delay in creating RCA/EDA analysis task (Table 4). As resourcing itself is not in the context of this thesis, improvement to control the amount and sharpen targeting of RCA/EDA requests is preferred action to better support CI/CD.

Sharper RCA/EDA selection criteria process. Improvement needed in RCA/EDA selection criteria to take the real customer impact and business value more into account. It is seen in the analysis that current RCA/EDAs lead to actual improvement actions too rarely (Figure 14) and even when improvements are done, those are not always done on areas from where faults to customers are leaking (Figures 17, 18, 19). A suggestion for improvement is to create a more systematic, data- and customer-oriented way of selecting cases requiring RCA/EDA to avoid non-value-adding analysis requests for software development teams. For example, when customer impact of the case is minor or negligible, RCA/EDA shall be rarely needed. In addition to this, improving the selection criteria will reduce the number of RCA/EDA analyses performed where no preventive actions have been found. As the target of RCA/EDA is always to improve the product, it is ineffective usage of resources if analysis without improvements is executed in high volume. In-stead, organization shall always concentrate in right and value adding topics (Latino & Latino & Latino 2011, chapter 1; Linders 2014).

Since CI/CD is already in place, internal bugs found during it are already findings that have been prevented from slipping to customers. It is indeed a primary goal for CI/CD to detect these defects before customer deliveries (Roy 2018; Bigelow 2021 b; Amazon Web Services b). Instead of

concentrating just to root causes of those faults found internally from CI/CD activities, there can be more benefit received when targeting internal RCA/EDA requests to those areas from where current and latest customer fault reports came. After all, RCA/EDAs are also done for these current customer cases, but since the nature of RCA/EDA is reactive, connecting new internal faults to new customer faults using different parameters and factors brings wider visibility and faster improvements. Before faults ending up to customers, it is not only important to fix them but ensure also systematic improvements are made to areas that already have a negative effect on customers today. It is an improvement on a larger scale. As systems are nowadays complex and complicated, when an individual RCA/EDA is concentrating to one root cause at the time, it might not find all the interaction and hidden patterns (Weeks 2018; Hammad 2022 c). It would also be a transition from a reactive way of working to a more proactive model when topics relevant to the customers are considered online. (Etti-Williams 2017; Hammad 2020 d).

Add visibility. Key item for agile software development is that online data is available to support decision making. One typical problem is that useful data is stored, but it is divided into separate applications, in worst case having different behavioural models and focusing only to specific software process areas. (Santos Jr et al. 2021, chapter Introduction.)

Connection between fault reports and RCA/EDA is existing in tools, but the overall pipeline view starting from the reported software defect to implemented improvement actions is missing. The whole lifecycle is not available and cannot be followed easily from current tools and dashboards. It is important to offer to the organization the possibility to follow statuses of all steps from one online source. The more there are separate tools connected into the workflow, the more important it is to have the visibility in one, easily accessible place. This problem with visibility was noticed during the data collecting phase in this research.

Another topic noticed is, that to mitigate the long lead time of internal RCA/EDAs, there could be own measurements added for that purpose. Currently only RCA/EDAs connected to customer reported faults, have lead time measured on company's level. RCA/EDAs for internal fault reports are not included in that measurement. This can be also one reason for longer lead times of internal fault report RCA/EDAs compared to customer reported ones (Figure 11).

5.3 How to left-shift RCA/EDA work?

As discussed in previous chapters, concentrating to internally reported faults and their RCA/EDAs would give the biggest benefit. The target shall be towards more predictive model instead of current reactive process. For left-shifting the RCA/EDA work, the biggest enhancement opportunity is seen on RCA/EDA triggering rules area of internally found defects. The solution preferred by this thesis to improve RCA/EDA request trigger practice is to take automated data-driven decision making into use, with online data connection between internal and customer originated fault reports. Nevertheless, that is not alone enough to increase predictivity and proactivity when the target is to avoid faults to be created at the first place. One rarely seen opportunity is to extend usage of RCA as a tool for identifying reasons for activities which went well and exceeded targets. Therefore, not only focusing on the things that cause problems, but to study, analyze and learn from cases where the goals were clearly achieved (Tableau 2018; Hammad 2020 c).

Data-driven decision-making bring efficiency and productivity. Pykes (2022) defines five areas which cannot be any more handled in modern companies without instant and accurate data. Those areas are decision making, problem solving, understanding performance, improving processes and understanding customers. The real benefit of data is received when it is gathered online, completely, accurately and it is connected to other relevant data. (Pykes 2022.) According to Taylor (2022 b), every second technology company used data-driven decision-making practices in 2020. The future trend of decision-making is predictive, but at the same time the trend also emphasizes the importance of efficiency and productivity in the decision-making process. Automated decision making and real-time available predictive reports will help organizations to reach enhancements on that area. (Capgemini 2023; Sevilla 2023.)

Learn from customers. One of the key elements of agile software development and CI/CD is learning from customers, short feedback cycles and customer insight, which can be enhanced by a data-driven decision-making process. (Berntsson Svensson & Feldt & Torkar 2019, 69; Santos Jr et al. 2021, chapter Introduction.) Decisions shall be based on online data and collected patterns and facts instead of intuition and observation. (Miller 2019). This means problems to be prioritized based on their impact to customers and the company. Organization shall always know how severe the problem is and how it is affecting to customer user experience when making decisions. (Latino & Latino & Latino 2011, chapter 1.)

As datasets connected to fault handling and RCA/EDAs are huge, it is not efficient to manage customer impact evaluation and related decisions manually, by humans and based on fixed rules, without full visibility to all patterns and factors online. As situations change fast in agile world, only connecting all relevant data sources together and base decisions on top of that data, will help to make RCA/EDAs more predictive and initiative-taking.

Left-shift RCA/EDA requests via data-driven decision making. Based on analysis results, RCAs are not completed fast enough, they do not always find improvement actions and faults are leaking to customers from certain areas. In this case, the mechanism must anticipate possible problem areas, correct the defects there and implement related improvements and preventive actions before customer delivery. For that purpose, RCA has been adopted in some companies as a tool for forecasting occurrence ratio of defects (Hammad 2020 d).

One solution to improve RCA practices to be more proactive and support CI/CD is to create an online connection between internal existing faults and open defects reported by customers, categorize those based on most crucial factors and patterns and to tune internal RCA/EDA requests based on that online data. This helps to focus on improving the areas that affect customers the most at that moment (Figure 20).

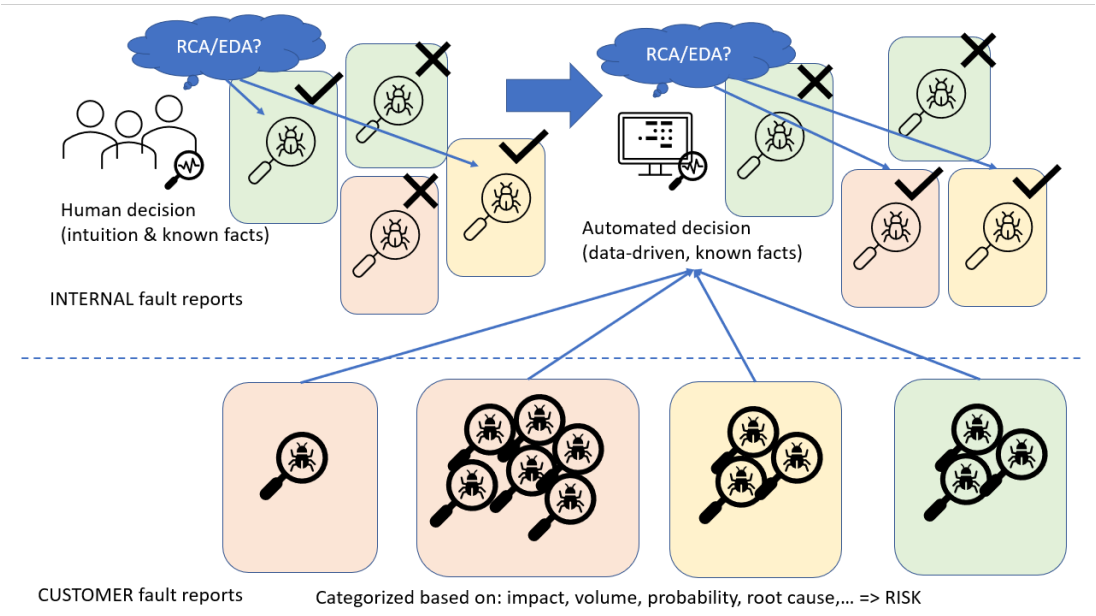


Figure 20. Proposal for decision process of RCA/EDA requests. Process to include the customer risk and business value evaluation by including data from latest customer fault reports. Transition from human manual decisions to automated, data-driven decision making.

Risk matrixes with categorized data is one possible solution. There are several parameters, which can be used to categorize fault reports: root cause/light root cause, customer impacts, recovery actions, occurrence ratio, number of similar fault reports connected, features (completed/faulty), the software area of the fault etc. In proposed solution both customer and internal faults will be categorized accordingly and probability to be used as one factor, which can be calculated for example based on number of open customer defects on that category (Figure 21). Actual parameters and threshold when RCA/EDAs are triggered and when not, can be defined by the organization, depending on collected data and its nature. Some categories can have more priority than others, but the most important thing is to connect the online customer impact and business value into the automated decision making.

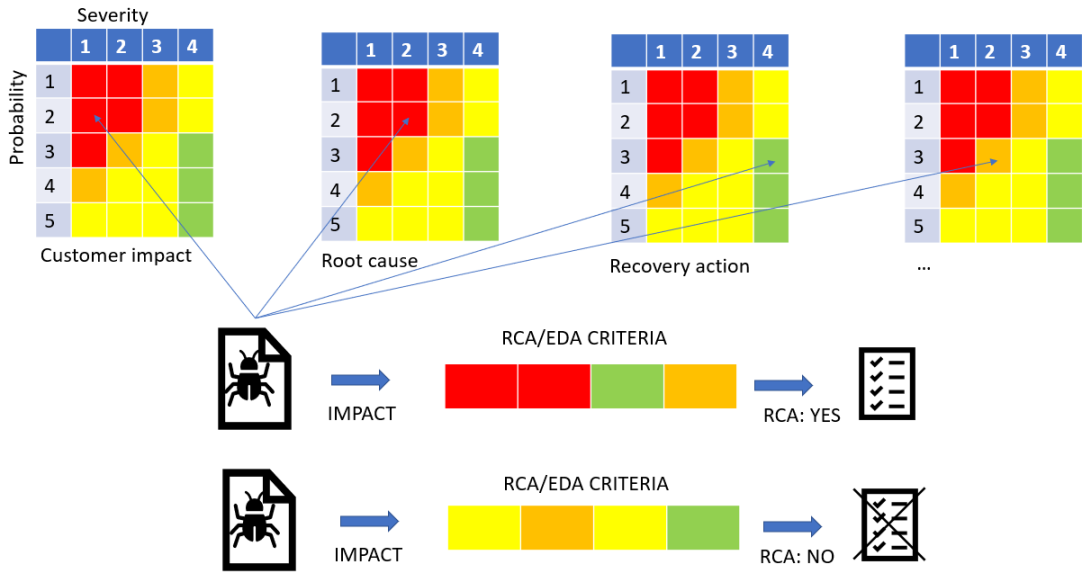


Figure 21. Evaluation of RCA/EDA request by connecting existing customer faults to decision making process.

Use RCA to analyse positive findings. There is also another, rarely used practice to support CI/CD by RCA process. Unforeseen opportunity is to use RCA as a tool to find out the root cause of succeeding topic. It would be real proactive usage of RCA to execute analysis to identify why certain areas or processes are performing well and reach their targets. According to Hammad (2020 c) identifying root causes and reasons behind those success stories, can lead to learnings and proactive organization level continuous improvements.

Similarly like to online visibility on areas causing troubles, it would be also possible to define online view to areas per-forming well based on fault handling related parameters. And that view could be used as a trigger for RCA request. It is one viable option for proactive usage of RCA process.

5.4 Research process

Research process of this thesis is discussed under this chapter. Four separate topics are highlighted.

The main challenge in the data-analysis of this research was related to combining different data-sources into one common data model. Data Portal tools as such are good tools when collecting and combining data from several different databases and tools into one individual location, form which users can then reach data via one interface. But as the data is already parsed and pre-formatted, it is not always serving exact needs for an end user. That is why there were difficulties in creating reliable relations between data tables from different queries in data model. Although the format of most important database keys, like software fault's report ID and RCA/EDA task ID, is unified, the data was stored into databases with the way the relationship between keys was not easy to build.

There were also feature and I&V databases used in the data model, but no findings which would have affected to research questions and results of this study found. That is why data from those databases was not reported in data-analysis. Findings and discussions in thesis are fully based on interview responses, RCA/EDA data and fault management data.

The idea of the study was that it can be repeated. Company's internal databases were used as a source and repeatability inside the company would be possible by another researcher. Detailed description about data sources in each visualization was shared in chapter three. Data model was planned and implemented so, that it would be easy to change for example time stamps in data queries to get online data from databases.

For this thesis, to be able to share results outside the company, graphs and visualizations were modified to hide company's confidential information. This caused some extra work, as original visualizations were not able to be used as such.

5.5 Research limitations

There are some known limitations in this study. The main limitation is related to the data. As it is case company's internal data, there is no access to source data by people outside the company.

The second limitation is related to data validity. Data accuracy was validated with most common techniques, outliers were analysed and removed or corrected when possible. But analysis also used data from drop-down fields, where values were selected manually by humans. Accuracy of those values was not able to be validated completely. There can be human mistakes or interpretation errors, like for example in selected root cause value or customer impact analysis value. Anyway, the dataset used in the research was so large, that trends are visible even individual values would have been selected wrongly.

Additionally, one important consideration regarding the data used in this thesis is that although RCA/EDA improvement tasks were not created for all RCA/EDA requests, this does not necessarily mean that these areas were not improved. The improvements could have been implemented via other separate tasks without direct contact with RCA/EDA. Although the RCA/EDA data used in this thesis does not always include improvement tasks for every software defect, it does not exclude implemented enhancements related to that area. In addition, as the basis of this thesis was frozen data from a certain predetermined time period, this analysis can only describe the RCA/EDA level at that point in time.

5.6 Future research suggestions

This research did not concentrate in actual troubleshooting and problem-solving phases. Those phases are crucial for software fault correction's overall lead time, but there was no statistically significant correlation seen in this research between fault report's age and RCA/EDA's age (see appendix 3). That is why fault report's lead time and troubleshooting area was left out from this study as the main objective was to improve RCA/EDA part from the software development

practices. But overall, to meet customer expectations in CI/CD, also troubleshooting phase has some enhancement options.

One area which is worth to investigate and invest in the future, is machine learning. For example, the initial root cause's identification via support of machine learning. When the problem in product is detected, machine learning could help to nail down the root cause faster compared to human made manual investigation and analysis. It could also lead to improvements on RCA/EDA area, where certain phases currently done manually, could be automated.

6 CONCLUSION

In this study, the aim was to identify how telecommunication software defect's root cause analysis (RCA) and escape defect analysis (EDA) could be moved in software development lifecycle to earlier phase than they currently are to better support continuous Integration (CI) and continuous delivery (CD) practices. RCA/EDA has been typically considered as a reactive process, which is triggered when a defect is found by the customer. But the practice has not been in use only for customer defects. It could be used also for organization's internally detected faults.

The problem statement was related to the hypothesis that current root cause analysis practice does not support CI/CD very well. According to analysis results this hypothesis was confirmed. And during the analysis phase of this study, it became clear that it was defects reported internally by organization and the related RCA/EDAs, that required more attention and improvement. Not all RCA/EDA practices are executed with the way which supports changed global environment, where agile software development practices are a norm and transition to CI/CD is ongoing. It means the traditional way to use RCA/EDA as a reactive tool based on escalation or fixed triggering rules is no more enough to meet customer expectations of fast delivery cycles and high software quality. Left-shifting the RCA/EDA work and transform from reactive to proactive model is mandatory.

There were three main pain points identified concerning internal fault reports and RCA/EDAs connected to them from CI/CD point of view. The first was long lead time of RCA/EDA tasks compared to RCA/EDAs from customer reported issues. Another problem was the substantial number of analyses where actual preventive actions had not been taken. This was connected also to missed analysis requests and preventive actions on certain areas from where faults were leaking to customers. Third topic was related to visibility. The whole pipeline from fault report creation to preventive action implementation was not easily seen, as separate tools were in use. There was no lead time metrics for internal fault report's related RCA/EDA. This also led to problems with prioritization and monitoring.

To get RCA/EDA to better support CI/CD practices like short delivery cycles, the lead time of RCA/EDAs shall be decreased. Other area to improve is to focus more on customer experience and value in RCA/EDA work. Areas that need more attention and from which defects are leaking to customers must be prioritized when making RCA/EDA request decisions. This is done by

focusing the RCA/EDA selection criteria more on real, online fact-based customer impact rather than pre-defined fixed rules, intuitive thinking or observations.

The goal of this thesis was to find ways how RCA/EDA work can be left-shifted in soft-ware development's life cycle to earlier phase. There were two topics identified how to transform RCA/EDA towards more predictive and proactive from current reactive mode. First issue proposed is to enhance automated data-driven decision-making mechanism to decide which internal fault reports shall get RCA/EDA. Current fixed rules with human-based manual evaluation and decision, can lead to RCA/EDAs requested and executed in areas which will not directly improve user experience and satisfaction. Customer view and impact might not enough effect on decisions. As datasets are huge, and online data usage mandatory, this work needs automation in decision making to take all aspects and parameters into account and improve RCA/EDAs overall efficiency. Customer view will be considered by connecting customer originated faults and their categories, like customer impact, recovery action, probability etc to internal fault reports and use for example risk matrixes to decide which internal open faults shall get an RCA/EDA request.

The final proposal is related to the RCA/EDA's nature. It is traditionally thought as a reactive process and connected always to defects and faults. But a finding can be also positive. There is unused potential in RCA process which can be used also analyzing topics which went well. When project or a team always meet their targets from fault handling parameters point of view, there is something done correctly there. That is why it would be essential for an organization to know the root cause of that success. When RCA with learnings would be executed to positive cases, it would be real proactive use of RCA process. And would lead to continuous improvements, efficiency and cost savings when good practices would be shared within the whole organization.

Though this study was planned for one company's needs and executed by using company's internal databases as a source, generalization of results is possible. Main topics handled by this thesis are globally known and similar type of data is collected and used by software development companies worldwide. And as well RCA/EDA and CI/CD processes and standards, like TL9000, are followed by all global telecommunication manufacturers. That is why generalization at least on some level is possible.

REFERENCES

Amazon Web Services a. What is Continuous Integration? Search date 15.04.2023. https://aws.amazon.com/devops/continuous-integration/.

Amazon Web Services b. What is Continuous Delivery? Search date 15.04.2023. https://aws.amazon.com/devops/continuous-delivery/

Anderson, Carl 2015. Creating a Data-Driven Organization. O'Reilly Media, Inc. Search date 15.12.2022. https://learning.oreilly.com/library/view/creating-a-data-driven/9781491916902/copyright-page01.html. Access required.

Barsalou, Mathew A. 2014. Root Cause Analysis. CRC Press. Search date 25.02.2023. https://learning.oreilly.com/library/view/root-cause-analysis/9781482258790/.

Bernstein, David -. What is Waste? AgileAlliance. Search date 03.03.2023. https://www.agilealliance.org/what-is-waste.

Berntsson Svensson, Richard & Feldt, Robert & Torkar, Richard 2019. The Unfulfilled Potential of Data-Driven Decision Making in Agile Software Development. International Conference on Agile Software Development, Agile Processes in Software Engineering and Extreme Programming XP2019 355 (5), 69 – 83. Search date 01.03.2023. https://link.springer.com/chapter/10.1007/978-3-030-19034-7_5.

Berteig, Mishkin 2020. Seven Options for Handling Interruptions in Scrum and Other Agile Methods. Search date 05.03.2023. https://berteig.com/how-to-apply-agile/seven-options-for-handling-interruptions-in-scrum-and-other-%E2%80%8Eagile-methods-3/.

Bigelow , Stephen J. 2021 a. How to handle root cause analysis of software defects. Search date 10.03.2023. https://www.techtarget.com/searchsoftwarequality/tip/How-to-handle-root-cause-analysis-of-software-defects.

Bigelow, Stephen J. 2021 b. CI/CD CI/CD pipelines explained: Everything you need to know. Search date 10.04.2024. https://www.techtarget.com/searchsoftwarequality/CI-CD-pipelines-explained-Everything-you-need-to-know.

Boeckman, Matthew 2017. There is No Root Cause: Emergent Behavior in Complex System. Search date 16.12.2022. https://youtu.be/EbcLmgrpLN4.

Brynjolfsson, Erik & Hitt, Lorin M. & Kim, Heekyung Hellen 2011. Strength in Numbers: How Does Data-Driven Decisionmaking Affect Firm Performance? SSRN. Search date 10.12.2022. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1819486.

Capgemini 2023. TechnoVision Report 2023, Executive Summary. Search date 05.03.2023. https://technovision.capgemini.com/right-the-technology/executive-summary.

Code[ish] 2021. Episode 114: Beyond Root Cause Analysis in Complex Systems. Search date 15.02.2023. https://dev.to/codeish/114-beyond-root-cause-analysis-in-complex-systems.

Cross, Aaron 2022. Why is Root Cause Analysis Important? Addressing Recurring Problems. Search date 03.03.2023. https://blog.thinkreliability.com/why-is-root-cause-analysis-important-addressing-recurring-problems.

Churchwell, Tom 2018. Escaped Defects. Search date 19.12.2022. https://www.leadingagile.com/2018/09/escaped-defects/.

Dawson, Maurice & Burrell, Darrell Norman & Rahim, Emad & Brewster, Stephen 2010. Integrating Software Assurance into the Software Development Life Cycle (SDLC). Journal of information systems technology & planning 3 (6), 49-53. Search date 12.03.2023. , https://www.researchgate.net/publication/255965523_Integrating_Software_Assurance_into_the_Software_Development_Life_Cycle_SDLC.

DiFrancesco, Bob 2021. Root Cause Analysis: The Opportunity for Proactivity. Search date 21.12.2022. https://www.armsreliability.com/page/resources/blog/root-cause-analysis-the-opportunity-for-proactivity.

Dunn, Sandy 2016. 5 tips for successful root cause analysis. Search date 10.03.2023. https://www.assetivity.com.au/articles/reliability-improvement/5-tips-for-successful-root-cause-analysis/.

Eclipse 2022. Root Cause Analysis: Meaning, Tools, Pitfalls and More. Search date 19.12.2022. https://www.eclipsesuite.com/root-cause-analysis/.

Edsel, Alexander 2016. Breaking Failure: How to Break the Cycle of Business Failure and Under-performance Using Root Cause, Failure Mode and Effects Analysis, and an Early Warning System. Pearson Packer. Search date 15.11.2022. https://learning.oreilly.com/library/view/breaking-failure-how/9780134387000/. Access required.

Etti-Williams, Jimmy 2017. What Is Root Cause Analysis (RCA) In A Business Environment? LinkedIn post. Search date 01.03.2023. https://www.linkedin.com/pulse/what-root-cause-analysis-rca-business-environment-etti-williams.

Gitential 2022. Gitential's Guide to the Cost of Fixing Bugs. Search date 19.03.2023. https://gitential.com/gitentials-guide-to-the-cost-of-fixing-bugs/.

GeeksforGeeks 2020. CI/CD: Continuous Integration and Continuous Delivery. Search date 14.04.2023. https://www.geeksforgeeks.org/ci-cd-continuous-integration-and-continuous-delivery/

Hammad, Madhuri 2020 a. Basic Principle of Root Cause Analysis. Search date 18.02.2023. https://www.geeksforgeeks.org/basic-principle-of-root-cause-analysis/.

Hammad, Madhuri 2020 b. Various RCA Techniques. Search date 18.02.2023. https://www.geeksforgeeks.org/various-rca-techniques/.

Hammad, Madhuri 2020 c. Advantages and Disadvantages of Root Cause Analysis. Search date 19.02.2023. https://www.geeksforgeeks.org/advantages-and-disadvantages-of-root-cause-analysis/.

Hammad, Madhuri 2020 d. Short note on Reactive and Proactive RCA. Search date 19.02.2023. https://www.geeksforgeeks.org/short-note-on-reactive-and-proactive-rca/.

Ismail, Nick 2017. Using data analytics to improve business processes and reduce waste. Search date 01.01.2023. https://www.information-age.com/using-data-analytics-improve-business-process-waste-4040/.

Kim, Gene & Humble, Jez & Debois, Patrick & Willis John 2016. The DevOps Handbook. IT Revolution Press. Search date 15.12.2022. https://learning.oreilly.com/library/view/the-devops-handbook/9781457191381/. Access required.

Latino, Robert J. & Latino, Kenneth C. & Latino Mark A. 2011. Root Cause Analysis, 4th Edition. CRC Press. Search date 29.12.2022. https://learning.oreilly.com/library/view/root-cause-analysis/9781439851272/. Access required.

Linders, Ben 2014. Success Factors for Root Cause Analysis in Software Development. Search date 11.03.2023. https://www.benlinders.com/2014/success-factors-for-root-cause-analysis-in-software-development/.

Miller, Kelsey 2019. Data-Driven Decision Making: A Primer for Beginners. Search date 11.01.2023. https://www.northeastern.edu/graduate/blog/data-driven-decision-making/.

Pykes, Kurtis 2022. The Importance of Data: 5 Top Reasons. Search date 18.01.2023. https://www.datacamp.com/blog/importance-of-data-5-top-reasons.

Roy, Arnab 2018. What Is Continuous Integration and Continuous Delivery? Search date 06.04.2023. https://dzone.com/articles/what-is-continuous-integration-andontinuous-delive.

Sacolick, Isaac 2022. What is CI/CD? Continuous integration and continuous delivery explained. Search date 09.04.2023. https://www.infoworld.com/article/3271126/what-is-CI/CD-continuous-integration-and-continuous-delivery-explained.html.

Santos Jr, Paulo Sergio & Perini Barcellos, Monalesa & de Almeida Falbo, Ricardo & Almeida, Joao Paulo A. 2021. From a Scrum Reference Ontology to the Integration of Applications for Data-Driven Software Development. ScienceDirect Information and Software Technology 136 (6) Article 106570. Search date 14.02.2023. https://doi.org/10.1016/j.infsof.2021.106570. Access required.

Sevilla, Manuel 2023. Process on the fly. Capgemini TechnoVision Report 2023. Search date 05.03.2023. https://technovision.capgemini.com/right-the-technology/trends-process-on-the-fly

Silverthorne, Valerie 2022. 10 Reasons why your business needs CI/CD. Search date 15.12.2022. https://about.gitlab.com/blog/2022/02/15/ten-reasons-why-your-business-needs-ci-cd/.

SoftwareTestingHelp 2023. Guide To Root Cause Analysis – Steps, Techniques & Examples. Search date 06.03.2023. https://www.softwaretestinghelp.com/root-cause-analysis/.

Tableau Software LLC. Root Cause Analysis Explained: Definition, Examples, and Methods. Search date 15.12.2022. https://www.tableau.com/learn/articles/root-cause-analysis.

Taylor, Petroc 2022 a. Amount of data created, consumed, and stored 2010-2020, with forecasts to 2025. Statista. Search date: 02.01.2023. https://www.statista.com/statistics/871513/worldwide-data-created/.

Taylor, Petroc 2022 b. Data-driven decision-making in global organizations 2020, by sector. Statista. Search date: 02.01.2023. https://www.statista.com/statistics/1235436/worldwide-data-driven-decision-making-organizations-by-sector/.

Tunguz, Tomasz & Bien, Frank 2016. Winning with Data. John Wiley & Sons, Inc. Search date 02.02.2023. https://learning.oreilly.com/library/view/winning-with-data/9781119257233/. Access required.

Ventiv Technology 2022. Blog post. Search date 29.12.2022. https://www.ventivtech.com/blog/whats-the-difference-between-internal-and-external-data.

Weeks, Derek 2018. DevOps: Improving Root Cause Analysis. Search date 16.12.2022. https://www.alldaydevops.com/blog/devops-improving-root-cause-analysis.

APPENDICES

APPENDIX 1: TL9000 problem report severity classification

APPENDIX 2: RCA/EDA analysis task structure in database and relations

APPENDIX 3: RCA/EDA time stamp correlation matrix

TL9000 PROBLEM REPORT SEVERITY CLASSIFICATION APPENDIX 1

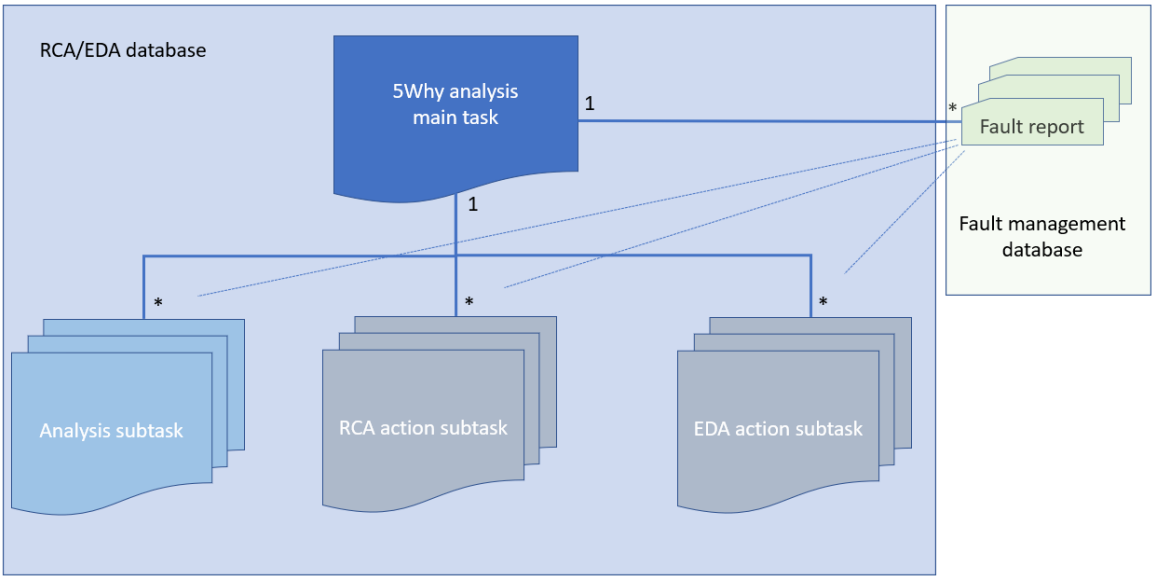
TL 9000 Quality Management System Measurement Handbook
https://tl9000.org/handbooks/measurements_handbook.html

1. **Critical:** Conditions that severely affect the primary functionality of the product and because of the business impact to the customer requires non-stop immediate corrective action regardless of time of day, or day of the week as viewed by a customer on discussion with the organization such as:
    - Product inoperability (total or partial outage),
    - A reduction in the capacity capability, that is, traffic/data handling capability, such that expected loads cannot be handled,
    - Any loss of emergency capability (for example, emergency 911 calls), or
    - Safety hazard or risk of security breach.

2. **Major:** Product is usable, but a condition exists that seriously degrades the product operation, maintenance, or administration, etc., and requires attention during pre-defined standard hours to resolve the situation.

    The urgency is less than in critical situations because of a less immediate or impending effect on product performance, customers, and the customer's operation and revenue such as:
    - Reduction in product's capacity (but still able to handle the expected load),
    - Any loss of administrative or maintenance visibility of the product and/or diagnostic capability,
    - Repeated degradation of an essential component or function, or
    - Degradation of the product's ability to provide any required notification of malfunction.
3. **Minor:** Other problems of a lesser severity than "critical" or "major" such as conditions that have little or no impairment on the function of the system.

RCA/EDA ANALYSIS TASK STRUCTURE IN DATABASE AND RELATIONS APPENDIX 2



There is always one main RCA/EDA 5Why analysis task. Under that task there can be several subtasks: analysis, RCA action, EDA action.

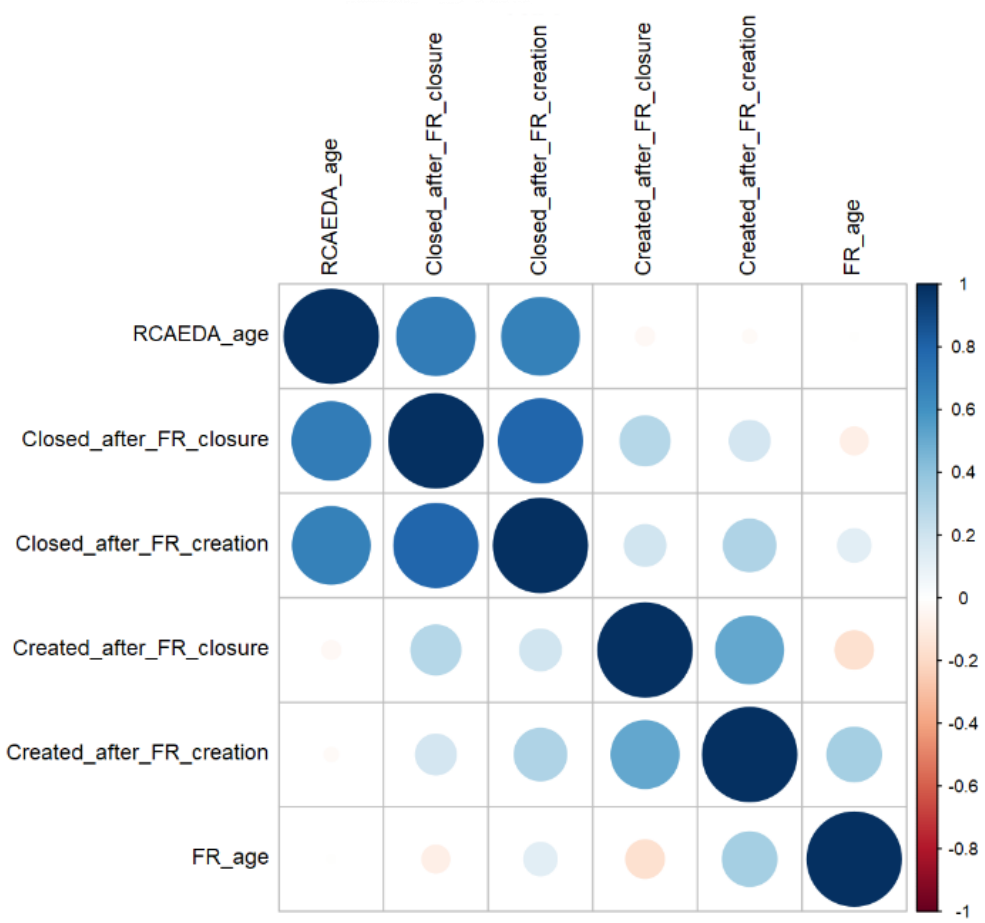
One Fault report can be connected in one main RCA/EDA task and several sub tasks.

RCA/EDA TIME STAMP CORRELATION MATRIX APPENDIX 3

Correlation coefficient for RCA/EDA time stamp related variables.

FR age = age of the Fault report connected to RCA/EDA action

RCAEDA age = age of the whole RCA/EDA, including analysis and improvement sub tasks



Calculation: Kendall's rank correlation, visualized by using R script editor in Microsoft's Power BI Reports